(19)日本国特許庁(JP)　(12)公開特許公報（A）　(11)特許出願公開番号

特開２００１－２４２９０６
（Ｐ２００１－２４２９０６Ａ）
(43)公開日　平成13年9月7日(2001.9.7)

(51)Int.Cl.[7]		識別記号	F I			テーマコード(参考)
Ｇ０５Ｂ	15/02		Ｇ０５Ｂ	15/02	Ｚ	
	11/36			11/36	Ｕ	
Ｇ０６Ｆ	3/14	３２０	Ｇ０６Ｆ	3/14	３２０ Ｃ	
	12/00	５１７		12/00	５１７	

審査請求　未請求　請求項の数２６　ＯＬ　外国語出願　（全１０７頁）

(21)出願番号　特願2000-358513(P2000-358513)

(22)出願日　平成12年10月18日(2000.10.18)

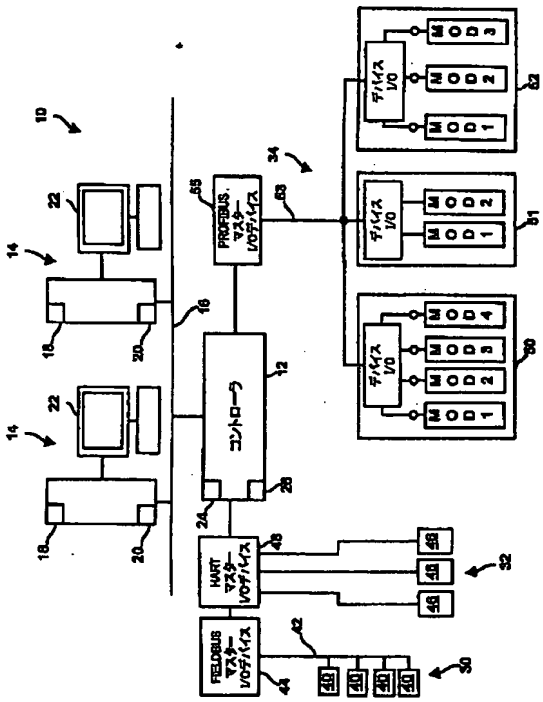
(31)優先権主張番号　09/420,182
(32)優先日　平成11年10月18日(1999.10.18)
(33)優先権主張国　米国（ＵＳ）

(71)出願人　594120847
フィッシャーローズマウント　システムズ，　インコーポレイテッド
アメリカ合衆国　78759　テキサス　オースティン　キャメロン　ロード　8301
(72)発明者　ハマック，　スティーヴン　ジェラード
アメリカ合衆国　78728　テキサス　オースティン　オカンナ　コート　14403
(72)発明者　ジュント，　ラリー　オスカー
アメリカ合衆国　78681　テキサス　ラウンド　ロック　ノースフィールド　305
(74)代理人　100065868
弁理士　角田　嘉宏　（外4名）

最終頁に続く

(54)【発明の名称】プロセス制御システム及び方法

(57)【要約】　（修正有）
【課題】プロセス制御システム及びその方法に関し、プロセスを監視するとともに、該プロセスへの変更を記録するシステム及び方法を提供する。
【解決手段】第１のデータベースは、プロセスの第１のコンフィギュレーションを表わす第１のデータを格納する一方、第２のデータベースは、プロセスの第２のコンフィギュレーションを表わす第２のデータを格納する。システムのコンフィギュレーション・ルーチンは、コンピュータでの読み取りが可能な記録媒体に格納され、プロセスの第１のコンフィギュレーションの変更を容易にするためにプロセッサによって実行されるように設定してある。システムのバージョン制御ルーチンは、コンピュータでの読み取りが可能な記録媒体に格納され、プロセスの第１のコンフィギュレーションの変更を示す第３のデータを第２のデータベースに格納するためにプロセッサによって実行されるように設定してある。



1

【特許請求の範囲】

【請求項１】　プロセスを制御するシステムにおいて、
コンピュータでの読み取りが可能な記録媒体と、
該コンピュータでの読み取りが可能な記録媒体と通信するプロセッサと、
前記プロセスの第１のコンフィギュレーションを表わす第１のデータを格納する第１のデータベースと、
前記プロセスの第２のコンフィギュレーションを表わす第２のデータを格納する第２のデータベースと、
前記コンピュータでの読み取りが可能な記録媒体に格納され、前記プロセスの第１のコンフィギュレーションの変更を容易にするために前記プロセッサによって実行されるように構成されたコンフィギュレーション・ルーチンと、
前記コンピュータでの読み取りが可能な記録媒体に格納され、前記プロセスの第１のコンフィギュレーションの変更を示す第３のデータを前記第２のデータベースに格納するためにプロセッサによって実行されるように構成されたバージョン制御ルーチンとを備えることを特徴とするプロセス制御システム。

【請求項２】　前記第１のデータベースは、前記プロセスの第１のコンフィギュレーションが現在のコンフィギュレーション・バージョンと対応するように、コンフィギュレーション・データベースを備えることを特徴とする請求項１記載のプロセス制御システム。

【請求項３】　前記第２のデータベースは、前記プロセスの第２のコンフィギュレーションが過去のコンフィギュレーション・バージョンと対応するように、バージョン制御データベースを備えることを特徴とする請求項１記載のプロセス制御システム。

【請求項４】　前記プロセスの第１及び第２のコンフィギュレーションは、第１及び第２の複数のプロセス・アイテムをそれぞれ有し、
前記プロセスの第１のコンフィギュレーションが、前記第１の複数のプロセス・アイテムのそれぞれのプロセス・アイテムのアイテム・コンフィギュレーションを含み、
前記プロセスの第２のコンフィギュレーションが、前記第２の複数のプロセス・アイテムのそれぞれのプロセス・アイテムのアイテム・コンフィギュレーションを含むように、
前記第１及び第２の複数のプロセス・アイテムのそれぞれのプロセス・アイテムは、対応するアイテム・コンフィギュレーションを有することを特徴とする請求項１記載のプロセス制御システム。

【請求項５】　前記バージョン制御ルーチンは、前記第３のデータを収集するために前記プロセスの第１のコンフィギュレーションの変更を監視することを特徴とする請求項１記載のプロセス制御システム。

【請求項６】　前記バージョン制御ルーチンは、チェックアウト／チェックイン手順を前記コンフィギュレーション・ルーチンに課すことによって変更を監視することを特徴とする請求項５記載のプロセス制御システム。

2

【請求項７】　前記チェックアウト／チェックイン手順は、自動であることを特徴とする請求項６記載のプロセス制御システム。

【請求項８】　プロセス・アイテムを有するプロセス制御システムにおいて、
プロセッサを有するコンピュータと、
前記アイテムのバージョンを確定するためにプロセッサによって実行されるように構成されたプロセス・コンフィギュレーション・アプリケーションと、
前記プロセス・アイテムのバージョンへの変更を記録及び制御するために前記プロセス・コンフィギュレーション・アプリケーションと通信するバージョン制御システムとを備えることを特徴とするプロセス制御システム。

【請求項９】　前記プロセス・コンフィギュレーション・アプリケーション及び前記バージョン制御システムは、前記プロセス・アイテムのバージョンへの変更の記録及び制御がユーザにトランスペアレントな方法で発生するように、統合されていることを特徴とする請求項８記載のプロセス制御システム。

【請求項１０】　前記変更は、自動化された方法で記録及び制御されることを特徴とする請求項８記載のプロセス制御システム。

【請求項１１】　前記バージョン制御システムは、前記プロセス・アイテムの前のバージョンを示すデータを格納することを特徴とする請求項８記載のプロセス制御システム。

【請求項１２】　前記バージョン制御システムは、前記プロセス・アイテムのバージョンと前記プロセス・アイテムの前のバージョンとの違いを表示するユーザ・インタフェースを提供する比較ツールを有することを特徴とする請求項１１記載のプロセス制御システム。

【請求項１３】　前記ユーザ・インタフェースは、視覚的に前記違いを表示することを特徴とする請求項１２記載のプロセス制御システム。

【請求項１４】　前記バージョン制御システムは、前記プロセス・アイテムのバージョンを前記プロセス・アイテムの前のバージョンと差し替えるためのロールバック・ツールを有することを特徴とする請求項１１記載のプロセス制御システム。

【請求項１５】　前記ロールバック・ツールは、前記プロセス・アイテムの何れかの下位アイテムが削除されたか否かについて自動的に判定することを特徴とする請求項１４記載のプロセス制御システム。

【請求項１６】　前記プロセス・アイテムを反映するデータがダウンロードされるプロセス・コントローラを更に備え、
前記バージョン制御システムは、前記プロセス・アイテ

ムを反映するデータが前記プロセス・コントローラにダウンロードされるときに、これを反映すべく、前記プロセス・アイテムのバージョンを示す情報を格納することを特徴とする請求項８記載のプロセス制御システム。

【請求項１７】　前記プロセス・アイテムを反映するデータが実行のためにダウンロードされるプロセス・コントローラを更に備え、

前記プロセス・コントローラにアクセスするプロセス・オペレータが現在実行されている前記プロセス・アイテムのバージョンについて知らされ得るように、前記データは、前記プロセス・アイテムのバージョンを示す情報を有することを特徴とする請求項８記載のプロセス制御システム。

【請求項１８】　プロセス・アイテムを有するプロセスを制御する方法において、

前記プロセス・アイテムの第１のバージョンを有する前記プロセスのコンフィギュレーションを確定するステップと、

前記プロセス・アイテムの第２のバージョンを生成するために、前記プロセスのコンフィギュレーションの変更を制御するステップと、

前記プロセスのコンフィギュレーションの変更に関連付けられた情報を記録するステップとを有することを特徴とするプロセス制御方法。

【請求項１９】　前記記録するステップは、トランスペアレントな方法で行われることを特徴とする請求項１８記載のプロセス制御方法。

【請求項２０】　前記制御するステップ及び前記記録するステップのうちの少なくとも１つは、自動化された方法で実行されることを特徴とする請求項１８記載のプロセス制御方法。

【請求項２１】　前記プロセス・アイテムの第１のバージョンと前記プロセス・アイテムの第２のバージョンとの違いを表示するユーザ・インタフェースを提供するステップを更に有することを特徴とする請求項１８記載のプロセス制御方法。

【請求項２２】　前記ユーザ・インタフェースは、視覚的に、前記違いを表示することを特徴とする請求項２１のプロセス制御方法。

【請求項２３】　前記プロセス・アイテムの第２のバージョンを前記プロセス・アイテムの第１のバージョンと差し替えるステップを更に有することを特徴とする請求項１８記載のプロセス制御方法。

【請求項２４】　前記差し替えるステップは、前記プロセス・アイテムの何れかの下位アイテムが削除されたか否かを判定するステップを有することを特徴とする請求項２３記載のプロセス制御方法。

【請求項２５】　前記プロセス・アイテムを反映するデータがプロセス・コントローラにダウンロードされたとき、これを反映すべく、前記プロセス・アイテムのダウンロードされたバージョンを示す情報を格納するステップを更に有することを特徴とする請求項１８記載のプロセス制御方法。

【請求項２６】　前記プロセス・コントローラにアクセスするプロセス・オペレータが前記プロセス・アイテムの現在のバージョンを知らされるように、前記プロセス・アイテムの現在のバージョンを示す情報をプロセス・コントローラにダウンロードするステップを更に有することを特徴とする請求項１８記載のプロセス制御方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本願発明は、一般には、プロセス制御システム及びその方法に関し、より詳しくは、バージョン制御を提供するために、プロセスを監視するとともに、該プロセスへの変更を記録するシステム及び方法に関する。

【０００２】

【従来の技術及び発明が解決しようとする課題】プロセス制御システムは、化学，石油，又はその他のプロセスに利用されるもののように、概して、アナログ及び／若しくはディジタル・バス又はその他の通信線又はチャネルを介して、少なくとも１つのホスト又はオペレータ・ワークステーションと一又は複数のフィールド・デバイスとに通信可能に結合される少なくとも１つの中央のプロセス・コントローラを備えている。前記フィールド・デバイス、即ち、例えば、バルブ，バルブポジショナー，スイッチ，及び送信機（例えば、温度，圧力，及び流量センサ）等は、バルブの開閉、及びプロセス・パラメータの測定の如きプロセスにおける機能を実行する。

【０００３】前記プロセス・コントローラは、フィールド・デバイス及び／又は入力／出力（I/O）装置を介した該フィールド・デバイスに関するその他の情報によってなされたプロセス計測を示す信号を受信し、制御ルーチンを実行するためにこの情報を使用して、当該プロセスの動作を制御すべく、前記入力／出力装置を介してバス又はその他の通信チャネルを通じて前記フィールド・デバイスへ送信される制御信号を生成する。前記フィールド・デバイス及び前記コントローラからの情報は、プロセスの現在の状態を見ること、プロセスの操作を変更すること、及びプロセスを設定すること等のプロセスに関するどの様な要求機能であってもオペレータが行えるように、前記オペレータ・ワークステーションによって実行される一又は複数のアプリケーションについて利用できるようになされている。

【０００４】幾つかのソフトウェア・ツールは、前記プロセスを設定する一方、変更することに関してオペレータを補助すべく開発されてきた。このようなツールは、前記フィールド・デバイス及びコントローラにより実行された各機能を表示する前記プロセスのグラフィカル形式の表示を、対応するアイテム又はオブジェクトとして

提供する。該アイテムは、例えば、特定のフィールド・デバイスによって実行される機能の全てがグループ化又はリスト化されるように、階層的な環境に整理され、示され得る。このアイテムは、該アイテムが前記プロセスにおける機能の関係で示されるように、制御テンプレート内で表示され得る。例えば、制御テンプレートは、線を相互接続することによって規定された入力及び出力の関係を有する複数のアイテムを表した、相互接続された一連のブロックを有した連続的なフローチャートから構成され得る。

【０００５】各アイテム（アイテム間の任意の入力又は出力の関係も含む）は、ソフトウェア・コンフィギュレーション・ツールに関連付けられたコンフィギュレーション・データベース内に格納されたデータによって定義される。データベース及びソフトウェア・ツールは、管理上の、プロセス・コンフィギュレーションの、及びその他の目的のためのデータにアクセスする多数のプロセス・オペレータ及びその他のユーザのための複数のワークステーションを主としてサポートするネットワークを介して利用可能になされている。しかし、例えば、或るオペレータが、別のオペレータのタスクに気付かなかったとき、又は前のコンフィギュレーション又は前記プロセスのバージョンへの戻り方を決められないほど余りにも多くの変更が生成されたときには、前記コンフィギュレーション・データベース内のデータの変更は、バージョン制御の問題に至ることがあり得る。例えば、本願出願人であるフィッシャー・ローズマウント・システムズ社から入手可能なDeltaV（商標）ソフトウェアを利用する従来のプロセス制御システムは、変更がなされた日付、及び該変更に責任があるユーザを示すデータを殆ど格納しないようになっている。このようなデータは、前記プロセスの前のバージョンのコンフィギュレーションを再構築することを、主としてユーザに可能とはさせない。

【０００６】これらのバージョン制御に関することは、コンフィギュレーション管理ツール（例えば、Rational社のClearCase及びMicrosoft社のSourceSafe（登録商標））によって、ソフトウェア開発の背景において言及されている。詳述すれば、これらの製品は、以後のデバッグ及び開発の労力を支援するために、ソフトウェア・ルーチンの開発について追跡，制御，及び管理する。最後に、コードの現在のバージョン及び過去のバージョンの両方を示すデータが格納される。しかし、このようなコンフィギュレーション管理ツールは、データのテキスト形式の表現に通常限られていて、主としてグラフィカル形式で記述される情報を格納及び表示することに適しているわけではない。

【０００７】グラフィカル形式のインタフェースは、計測システムに関連してプログラミングを容易にするためにNational Instruments社によって開発されている。National Instruments社のインターネット・ウェブサイト（www.ni.com）から集められた情報によれば、「LabVIEW」という名前で市場に出回っている製品は、グラフィカル形式のプログラミング言語（このグラフィカル形式のプログラミング・インタフェースをサポートするG言語）を利用している。このウェブサイトは、さらに、「LabVIEW」という製品が、プログラマーに、視覚的にG言語で記述されるコードの２つのファイル間の違いを比較させる開発ツールを備えていると記述している。

【０００８】本願発明は、プロセスの前記コンフィギュレーションの変更を示すデータを、前記変更に責任があるユーザのアイデンティティー，前記変更の時間及び日付，並びに前記変更の背景となる理由等の、前記変更に関する情報とともに記録するためのプロセス制御システム及び方法を提供することを目的とする。

【０００９】

【課題を解決するための手段】本願発明の１つの観点によれば、プロセスを制御することに役立つシステムは、コンピュータでの読み取りが可能な記録媒体と、該コンピュータでの読み取りが可能な記録媒体と通信するプロセッサと、第１及び第２のデータベースとを備える。該第１のデータベースは、前記プロセスの第１のコンフィギュレーションを表わす第１のデータを格納し、前記第２のデータベースは、前記プロセスの第２のコンフィギュレーションを表わす第２のデータを格納する。本願発明のシステムは、コンフィギュレーション・ルーチン及びバージョン制御ルーチンを更に有し、両者は、コンピュータでの読み取りが可能な記録媒体に格納され、前記プロセッサによって実行されるように構成される。前記コンフィギュレーション・ルーチンは、前記プロセスの第１のコンフィギュレーションの変更を容易にし、前記バージョン制御ルーチンは、前記プロセスの第１のコンフィギュレーションの変更を示す第３のデータを第２のデータベースに格納する。

【００１０】好ましい実施形態によれば、前記第１のデータベースは、前記プロセスの第１のコンフィギュレーションが現在のコンフィギュレーション・バージョンと対応するように、コンフィギュレーション・データベースを有する。前記第２のデータベースは、前記プロセスの第２のコンフィギュレーションが過去のコンフィギュレーション・バージョンと対応するように、バージョン制御データベースを有することが好ましい。

【００１１】別の好ましい実施形態によれば、前記プロセスの第１及び第２のコンフィギュレーションは、第１及び第２の複数のプロセス・アイテムをそれぞれ有する。該第１及び第２の複数のプロセス・アイテムのうちの各プロセス・アイテムは、前記プロセスの第１のコンフィギュレーションが前記第１の複数のプロセス・アイテムのうちの各プロセス・アイテムのアイテム・コンフィギュレーションを有するように、対応するアイテム・

コンフィギュレーションを有し、また、前記プロセスの第２のコンフィギュレーションは、前記第２の複数のプロセス・アイテムのうちの各プロセス・アイテムのアイテム・コンフィギュレーションを有する。

【００１２】さらに別の好ましい実施形態によれば、前記バージョン制御ルーチンは、前記第３のデータを収集するために前記プロセスの第１のコンフィギュレーションの変更を監視する。前記バージョン制御ルーチンは、チェックアウト／チェックイン手順をコンフィギュレーション・ルーチンに課すことによって前記変更を監視することが可能である。換言すれば、前記チェックアウト／チェックイン手順は自動である。

【００１３】本願発明の別の観点によれば、プロセス・アイテムを有するプロセス制御システムは、プロセッサを有するコンピュータと、前記アイテムのバージョンを確定するためにプロセッサによって実行されるように構成されたプロセス・コンフィギュレーション・アプリケーションと、前記プロセス・アイテムのバージョンへの変更を記録及び制御すべく前記プロセス・コンフィギュレーション・アプリケーションと通信するバージョン制御システムとを備えている。

【００１４】本願発明の更に別の観点によれば、プロセス・アイテムを有するプロセスを制御する方法は、前記プロセス・アイテムの第１のバージョンを有する前記プロセスのコンフィギュレーションを確定するステップと、前記プロセス・アイテムの第２のバージョンを生成する前記プロセスのコンフィギュレーションの変更を制御するステップと、前記プロセスのコンフィギュレーションの変更と関連する情報を記録するステップとを有する。

【００１５】

【発明の実施の形態】以下、本願発明の実施形態を図面を参照しながら説明する。

【００１６】プロセスには、概して多くの継続的な設計及び開発を必要とする。その結果として、前記プロセスのコンフィギュレーションの如何なる変更をも示すデータを記録することは、多くの場合には望ましいことである。例えば、このようなコンフィギュレーション情報は、政府機関によって監視される一方、製薬業界で利用されている。前記プロセス（前記プロセス制御システムも同様）がより複雑になるにつれて、前記プロセスの前のバージョンのコンフィギュレーションを正確に確定することがより面倒となる。

【００１７】本願発明は、プロセスの前記コンフィギュレーションの変更を示すデータを、前記変更に責任があるユーザのアイデンティティー，前記変更の時間及び日付，並びに前記変更の背景となる理由等の、前記変更に関する情報とともに記録するためのシステム及び方法を提供する。本願発明のシステム及び方法が、ユーザに、任意の２つのバージョンの違いを比較させ、また、所望の前のコンフィギュレーション・バージョンに前記プロセスを戻させることを可能にするように、前記データは、本願発明に従って前記プロセス・コンフィギュレーションのバージョンとして格納される。

【００１８】さて、図１を参照して、プロセス制御システム10は、通信ネットワーク16（例えば、イーサネット（登録商標）接続等）を介して、一又は複数のホスト・ワークステーション又はコンピュータ14（任意の種類のパーソナル・コンピュータ，ワークステーション等）に接続されたプロセス・コントローラ12を備えている。前記ホスト・ワークステーション14の各々は、プロセッサ18，メモリ20，及びディスプレイ22を備える。同様に、前記プロセス・コントローラ12は、ここでは例示だけに留めておくが、本願出願人であるフィッシャー・ローズマウント・システムズ社により販売されている「DeltaV（商標）」コントローラは、プロセッサ24と、プロセスの制御を実行するために該プロセッサ24で使用されるプログラム，制御ルーチン，及びデータを格納するメモリ26とを備えている。前記プロセス・コントローラ12は、異なるデバイス・ネットワーク（例えば、Fieldbusデバイス・ネットワーク30，HARTデバイス・ネットワーク32，及びProfibusデバイス・ネットワーク34を含む）内で多数のフィールド・デバイスに通信ネットワーク16を介して結合されている。勿論、前記プロセス・コントローラ12は、図１に示された前記デバイス・ネットワークに加えて又はその代わりに、4～20 mAのデバイス・ネットワーク及びその他のローカル又はリモートI/Oデバイス・ネットワークのようなその他の種類のフィールド・デバイス・ネットワークに接続され得る。前記プロセス・コントローラ12は、そこに格納され又はそれと関連する一又は複数のプロセス制御ルーチンを実行又は監視し、デバイス・ネットワーク30，32，及び34内のデバイスと、ホスト・ワークステーション14と通信して、一又は複数のユーザ，オペレータ，プロセス設計者等に前記プロセスに関連する情報を提供するために、前記プロセスを制御する。

【００１９】典型的な事例において、前記Fieldbusデバイス・ネットワーク30は、Fieldbusマスター I/Oデバイス44（一般には「リンク・マスター・デバイス」と称される）にFieldbusリンク42を介して接続されたFieldbusデバイス40を備え、ローカル接続を通じてプロセス・コントローラ12に接続されている。同様に、前記HARTデバイス・ネットワーク32は、標準的なローカル・バス又はその他の通信線を介して前記プロセス・コントローラ12に接続されたHARTマスターI/Oデバイス48に通信線を介して接続された多数のHARTデバイス46を備えることが可能である。前記Profibusデバイス・ネットワーク34は、Profibusマスター I/Oデバイス55にProfibusリンク又はバス53を介して接続された３つのProfibusスレーブ・デバイス50，51，及び52を備えているものとして図示され

ている。前記Profibusマスター I/Oデバイス55は、標準的なI/Oインタフェース・カードに取り付けられたProfibusPCMCIA（Personal Computer Memory Card International Association）カードの形態とすることが可能である。

【００２０】一般的に言って、図１のプロセス制御システム10は、バッチ・プロセスを実行するために使用され得る。これは、例えば、前記ホスト・ワークステーション14又はプロセス・コントローラ12のうちの１つが、バッチ・エグゼクティブ・ルーチンを実行するものであって、該ルーチンは、製品（例えば、食品又は薬品）を生産するために必要である一連の異なるステップ（一般には「フェーズ」と称される）を実行するために、一又は複数の前記フィールド・デバイス（恐らくその他の設備も同様である）の操作を指示する高水準の制御ルーチンである。異なるフェーズを実行するために、前記バッチ・エグゼクティブ・ルーチンは、実行されるべきステップ、或いはステップ，該ステップの順序等と関連する量及び時間を特定する、一般に「レシピ」と称されるものを使用する。そして、１つのレシピに対するステップは、例えば、原子炉容器（図示せず）を適切な材料で満たし、原子炉容器内の材料を混ぜ、或る時間で或る温度まで原子炉容器内の材料を熱し、原子炉容器を空にし、それから、次のバッチ処理の準備をすべく原子炉容器をきれいにすることを含むかも知れない。これらのステップの各々は、バッチ処理の対応するフェーズを規定することが可能であり、前記プロセス・コントローラ12内のバッチ・エグゼクティブ・ルーチンは、前記フェーズの各々のために異なる制御アルゴリズムを実行する。勿論、特定の材料，材料の量，加熱温度，及び時間等は、レシピに応じて異なり、従って、これらのパラメータは、製造又は生産される製品及び使用されるレシピに応じて、或るバッチ処理から別のバッチ処理へ変更されることが可能である。

【００２１】また、前記プロセス制御システム10は、バッチ処理の一部として開始されるものに加えて事実上連続するプロセス操作の実行機能を備えることも可能である。このように、ここで使用されるように、前記「プロセス」は、前記プロセス制御システム10によって実行又は実施される任意の数のバッチ及び／又は連続するプロセス操作であるものとして理解され、また、これらを含み得ると理解されるべきである。前記プロセスの実行中においては、前記プロセス制御システム10は、「ランタイム・モード」であると言える。前記プロセス制御システム10が操作のコンフィギュレーション・モードにある間、ランタイム中に利用されるべき前記パラメータ及び制御アルゴリズムが規定される。前記コンフィギュレーション・モードの間（プロセス操作が実行されている期間と場合によっては重なることがあり得る）、一又は複数のホスト・ワークステーション14上で実行される一又は複数のソフトウェア・アプリケーションは、ユーザに、前記パラメータ，制御アルゴリズム等を指定することを可能にするために利用され、そのようにすることにより、一般には、ユーザが前記プロセスのコンフィギュレーションを設計することを可能にする。

【００２２】さて、図２を参照して、前記プロセス制御システム10は、一又は複数のプロセス・コンフィギュレーション・アプリケーション（例えば、「Control Studio（商標）」及び「Recipe Studio（商標）」であり、これらは、両方とも本願出願人であるフィッシャー・ローズマウント・システムズ社から入手可能である）に基づいたプロセス・コンフィギュレーション・システムを備えている。前記プロセス・コンフィギュレーション・アプリケーション（これはソフトウェアによって実行されることがあり得る）は、バッチ又は連続するプロセス操作の何れか（以下、全体に亘って「前記プロセス」と称す）又はその或る部分を設計するために利用される。図２は、ユーザが前記プロセス（プロセス制御システム）10を設定することを可能にする典型的なプロセス・コンフィギュレーション・アプリケーションのメイン制御ウィンドウの画面表示である。図２のプロセス・コンフィギュレーション・アプリケーションは、通信ネットワーク16内の中央のサーバ（図示せず）に対応し得るホスト・ワークステーション14の任意の１つで実行されることが好ましい。これに代えて、前記アプリケーションは、複数のホスト・ワークステーション14が実行を担うような分配方法にて実行することも可能である。

【００２３】一般に、前記プロセス・コンフィギュレーション・アプリケーションは、図２に示された前記メイン制御ウィンドウを有するユーザ・インタフェースを提供する。前記メイン制御ウィンドウは、テキスト形式のプルダウン式のコマンド・メニュー60，図形的なコマンド・バー62，機能ライブラリ・フレーム64，及びフロー・ダイアグラム・フレーム66を有している。前記ユーザ・インタフェースのこれらの部分の各々は、典型的なウィンドウズ（登録商標）の操作手法に従って大きさを設定され、又は配置されることが可能である。複数のアイコン68が機能ライブラリ・フレーム64内に表示され、その各々が前記フロー・ダイアグラム・フレーム66内で生成された一連のフローチャートに組み込まれるべき、対応する機能ブロックを表わす。前記一連のフローチャートは、例えば、設計されるプロセスのフェーズ中に実行されるべき制御プログラムを規定することが可能である。前記制御プログラムを構築するには、ユーザが、例えば、ポインティング・デバイス（マウス等）を用いて、機能ライブラリ・フレーム64内の所望するアイコンを選択することによって一又は複数のアイコン68をアクティブにし、このアクティブにしたアイコンを前記フロー・ダイアグラム・フレーム66へドラッグする。それから、前記プロセス・コンフィギュレーション・アプリケ

ーションは、前記フロー・ダイアグラム・フレーム66内に、追加（add）ブロック70（これは機能ブロックを表わす）のようなオブジェクトを生成する。前記フロー・ダイアグラム・フレーム66におけるブロック70と関連するデータは、該ブロック70が入力ポート，出力ポート等を有するか否か（示されている前記ブロックの一部として視覚的に表わされる）とともに、実行される機能を規定する。前記一連のフローチャート（例えば、入力パラメータ）に追加された機能ブロック及びその他のオブジェクトは、前記オブジェクト間のデータ関係を確立するために前記フロー・ダイアグラム・フレーム66において、ユーザによって（例えば、コマンド・バー62から選択された描画ツールを用いて）描かれた線で結合される。また、各機能ブロックが実行されるべき時間系列（順序）に関する情報は、ユーザによって指定されることが可能であり、また、表示されたオブジェクトと関連付けて示されることが可能である。

【００２４】各機能ブロック，フェーズ等を表わすデータは、コンフィギュレーション・データベースに格納され、プロセス・コンフィギュレーション・アプリケーションによってアクセス及び展開される。本願発明の実施に際しては、如何なる特定の種類のデータベースにも限定されるものではなく、また、前記コンフィギュレーション・データベースは、如何なる形態又は構造をなしてあってもよい。例えば、前記コンフィギュレーション・データベースに格納されたデータは、ローカルに、又は前記コンフィギュレーション・データベースに格納されるデータが通信ネットワーク16に亘って分配されるように局所的に格納される必要はない。しかし、前記コンフィギュレーション・データベースは、前記プロセス・コンフィギュレーションを表わすデータ（例えば、前記フロー・ダイアグラム・フレーム66で設計される一連のフローチャート内に割り当てられた機能ブロック間の関係）を格納するホスト・ワークステーション14のうちの１つのメモリ20に配置されたオブジェクト指向のデータベースを備えていることが好ましい。一旦、前記一連のフローチャートが上述の機能性を使用して設計される場合、結果として生じる制御プログラムは、前記プロセス・コンフィギュレーションの合成アイテムとして格納されることが可能である。「Control Studio（商標）」ソフトウェア・システムによって開発された合成アイテムは、同様に、モジュール（図中「MOD」で示す）を構築すべく利用されることが可能である。また、そのために、モジュールを表わすデータを、前記コンフィギュレーション・データベースに格納することも可能である。合成アイテム及びモジュールの両方を、一連のフローチャートを使用して表わすことが可能である。

【００２５】一連のフローチャートを使用して表わされ得るアイテムの更なる例は、レシピ，手順，及び単位手順を含む。これらのアイテムは、前記プロセスの一部として実行されるべきバッチ処理（又はその一部）を表わすプロセス・コンフィギュレーション・システム（「Recipe Studio（商標）」アプリケーション）の別の部分によって設計されることが可能である。前記「Recipe Studio（商標）」アプリケーションは、図２に示されるものと同様のユーザ・インタフェースを生成し、機能ブロックとともに、モジュールを表わすアイコンを有したライブラリ・フレームを提供することが可能である。それから、前記レシピ，手順，及び単位手順を表わすデータは、前記コンフィギュレーション・データベースに格納されることも可能である。

【００２６】これとともに、これらの機能ブロック，モジュール，レシピ等を表わすデータは、それが前記プロセス制御システム10によって現在実行されているように、前記プロセスを規定することが可能である。このために、前記コンフィギュレーション・データベースは、指示が前記プロセス・コントローラ12及びフィールド・デバイスにダウンロードされたときに、アクセス及び利用される。

【００２７】さて、図３を参照して、コンフィギュレーション・データベースに格納されたデータは、任意のデータベース管理上のインタフェースが利用され得る理解とともに、以下に「Explorerシステム」と称するコンフィギュレーション・データベース管理上のインタフェース（例えば、「DeltaV（登録商標）Explorer」）を介して、ユーザに提示され得る。前記Explorerシステムは、階層の要素を変更するためのコンフィギュレーション・ツールの組を有するウィンドウ型の環境におけるコンフィギュレーション階層を記述する。詳述すれば、前記Explorerシステムによって展開されたメイン制御ウィンドウ80は、前記コンフィギュレーション階層が表示される階層フレーム82と、前記階層の各コンポーネントのための追加情報が表示されるコンテンツ・フレーム84とを有する。前記階層フレーム82は、ライブラリ・フォルダ88及びシステム・コンフィギュレーション・フォルダ90を表わす２つの２番目の階層とともに、前記階層（例えば、プロセス・コンフィギュレーション全体を表わす最上位層86）の様々なレベルを識別する複数のアイコンを有している。前記ライブラリ・フォルダ88は、機能ブロック及び／又はモジュールが制御プログラムの一部としてフェーズ又はプロセス内で適用されるプロセス設計の間、前記情報が利用されることを含み得る。それから、前記システム・コンフィギュレーション・フォルダ90は、その中の一又は複数のフォルダ（例えば、レシピ・フォルダ92）におけるデザイン手順の結果を含む。前記２番目の階層を超える各フォルダは、公知のウィンドウズ手法に従って、前記階層フレーム内で拡大又は縮小されることが可能である。これに代えて、フォルダの前記コンポーネントは、ポインタ等を使用して、前記フォルダの名前を選択することによってコンテンツ・フレーム

84内に表示されることが可能である。例えば、「AREA A」と関連する制御プログラム「LOOP」のコンテンツは、コンテンツ・フレーム84内でその名前が識別される多数のアイテム又はオブジェクトを利用する。これらのアイテムのうちの１つを選択することは、前記アイテムのプロパティに関する情報（例えば、オブジェクトの種類，前記アイテムを展開するために使用されるアプリケーション，データ・ストレージの位置等）を、プロセス・コンフィギュレーション・アプリケーション（例えば、「Control Studio」）一連のフローチャートのグラフィカル形式の表示とともに、また、一連のフローチャートが存在しない場合、テキスト表示とともに、ユーザに提供することが可能である。

【００２８】図２及び図３にそれぞれ関連付けて説明された前記プロセス・コンフィギュレーション・アプリケーション（例えば、「Control Studio（商標）」及び「Recipe Studio（商標）」）及び前記Explorerシステムによって提供されたツールは、任意の望ましい範囲において単一のアプリケーションに統合され得る点を注記しておく。さらに、前記プロセスのコンフィギュレーションが達成される方法は、本願発明の実施には関係がない。このように、説明の簡略化のために、以下に使用されるように、前記「コンフィギュレーション・アプリケーション」は、図３に関連付けて説明される前記Explorerシステムとともに、図２に関連付けて説明される前記プロセス・コンフィギュレーション・アプリケーションを含むと理解されるべきである。

【００２９】図４を参照して、ユーザ・インタフェース94は、ユーザに一又は複数のコンフィギュレーション・アプリケーション96を実行させることを可能にするために、前記ホスト・ワークステーション14（図１参照）のうちの１つのディスプレイ22上に生成される。本願発明によれば、前記ユーザ・インタフェース94は、バージョン制御及び監査証跡システム98（以下、「VCATシステム」と称す）と関連しても生成され、一般に、前記プロセスのコンフィギュレーションに関する履歴情報を記録及び管理するために前記コンフィギュレーション・アプリケーション96と連動するようになっている。前記コンフィギュレーション・アプリケーション96及びVCATシステム98の両方は、上述したように、コンフィギュレーション・データベース100にアクセスして通信を行い、前記プロセスの現在のコンフィギュレーションを表わすデータを格納する。上記VCATシステム98は、バージョン制御データベース102とも通信し、本願発明に従って管理される。

【００３０】前記バージョン制御データベース102は、前記プロセス・コンフィギュレーションで利用される各アイテムの任意の数の前のバージョンを示すコンフィギュレーション履歴データを有している。これと組み合わせて、前記アイテムの全てのための履歴データは、前記プロセスの過去のコンフィギュレーションを再構築するために使用されることが可能である。詳述すれば、前記コンフィギュレーション・データベース100（該コンフィギュレーション・データベース100にもはや存在しないものとともに）における各アイテムについて、そのアイテムのコンフィギュレーションを表わすデータとして複数のバージョンが格納される。例えば、或るアイテムが、それが生成された時から、３回変更されているものとする。前記バージョン制御データベース102は、従って、生成の時点でのこのアイテムのコンフィギュレーションを示すデータ、即ち「バージョン1」を有する。また、前記バージョン制御データベース102は、３回の変更の各々の後におけるこのアイテムのコンフィギュレーションを示すデータに対応する「バージョン2」，「バージョン3」，及び「バージョン4」も有している。

【００３１】前記コンフィギュレーション履歴データは、このように、機能ブロック，モジュール，フェーズ，レシピ，及び前記プロセス・コンフィギュレーションのその他の任意の観点からなされる変更の全てについて表わしている。前記変更は、コンフィギュレーション・アプリケーション96を使用してなされることがあり得るが、必須ではない。しかし、このような場合には、本願発明の実施においては、図４で概略的に示されるように、また、以下に更に詳細に説明されるように、前記コンフィギュレーション・アプリケーション96によって生成されたユーザ・インタフェース94に、上記VCATシステム98の機能性を統合することによって好ましくは達成される。このために、前記コンフィギュレーション・アプリケーション96及びVCATシステム98は、単一の、統合システムとして組み合わせることが可能であるが、必須ではない。より明確にするために、しかし、本願発明に従って実行されるタスクは、前記コンフィギュレーション・アプリケーション96及びVCATシステム98に各別に帰する。

【００３２】上記VCATシステム98は、ホスト・ワークステーション14の一又は複数を使用し、前記プロセス・コンフィギュレーションへの如何なる変更についても監視を許容するように、実行されることが好ましい。このような監視は、上述した前記統合システムによって助成され得る。これに代えて、上記VCATシステム98は、オペレータ又はプロセス設計者によって利用されるホスト・ワークステーション14と対応しないワークステーション又はその他のデバイス上で実行されることが可能であるが、前記コンフィギュレーションの変更を示す記録データのために、プロセス制御システム10と通信する。

【００３３】前記データベース100，102の一方又は両方におけるデータは、例えば、ホスト・ワークステーション14のうちの１つに関連付けられたメモリ20又は磁気的な若しくは光学的な記録媒体の如き、前記プロセス制御システム10内の任意の物理的な位置に配されたコンピュ

ータでの読み取りが可能な記録媒体に格納されることが可能である。これに代えて、前記データベース100，102の一方又は両方は、これに格納されたデータに、前記ホスト・ワークステーション14がネットワーク（例えば、イントラネット，インターネット，又はその他の任意の通信媒体）を介してアクセスするように、リモート位置に格納されることが可能である。さらに、各データベース100，102に格納されたデータは、同一のコンピュータでの読み取りが可能な記録媒体に格納される必要はない。そして、データベース100，102の何れかの任意の部分が、その他の部分を格納するデバイス又はメディアとは別であるメモリ・デバイス又は記録媒体に格納されることも可能である。

【００３４】図４において、上記VCATシステム98は、前記バージョン制御データベース102とは、識別が明確な別個のものとして示されている。その他の実施形態において、前記バージョン制御データベース102は、上記VCATシステム98の一部を形成する。同様に、前記コンフィギュレーション・データベース100及びバージョン制御データベース102は、個別で識別の明確なデータ構造を構成することが可能であるが、必須ではない。即ち、前記データベース100，102は、同一の記録媒体に配置されることがあり得、実際、前記プロセス制御システム10に貢献する共通のデータベースの一部を構成することが可能である。従って、ここで使用される「データベース」は、如何なる特定のデータ構造にも限定されるものと理解されるべきではない。

【００３５】１つの実施形態においては、前記バージョン制御データベース102は、リレーショナル・データベースを備えている。これに代えて、前記バージョン制御データベース102は、バージョン制御ツール（例えば、Microsoft社の「Visual SourceSafe（登録商標）」）を提供するリファレンス・アプリケーションを使用し、バージョン制御データのストレージ（貯蔵所）として生成されることが可能である。さらに別の実施形態において、前記バージョン制御データベース102は、ファイル・ベースのものであることが可能である。

【００３６】図５は、図３に示したものと同様のユーザ・インタフェース94を示す画面表示であって、前記コンフィギュレーション・アプリケーション96と統合されたような、上記VCATシステム98によって提供される機能性を取り入れたものを示している。このような統合システムでは、一又は複数のツールが、前記プロセス制御システム10の管理の状況における上記VCATシステム98の実行及び制御のためのコンフィギュレーション・アプリケーション96に付与されている。例えば、上記VCATシステム98が有効とされ（以下に更に詳細に説明される）、ユーザがダブルクリック又はそれを選択することによって、コンフィギュレーション・アプリケーション96で管理されているアイテムを表示及び／又は変更しようとすると、この先へ進む前に、コンフィギュレーション・データベース100から前記アイテムの「チェックアウト」に必要性についてユーザに警告するダイアログ110が、ユーザ・インタフェース94内に生成される。同様に、ユーザが新しいアイテムをプロセス・コンフィギュレーションに追加するとき、このような統合システムは、新しいアイテムを、バージョン制御データベース102及びコンフィギュレーション・データベース100の両方に自動的に追加し、コンフィギュレーション・アプリケーション96内でその生成を容易にするために前記アイテムをチェックアウトする。

【００３７】要するに、上記VCATシステム98は、前記プロセスの設計に対して、チェックアウト／チェックイン環境を課すことが好ましい。この環境においては、上記VCATシステム98が前記アイテムに関する情報を記録することが可能であり、そのようにする際に、これから行う「チェックイン」操作に備えることが可能であるように、前記アイテムは、変更がなされ得る前に、チェックアウト（手動又は自動で）されなければならない。もし、例えば、コンテンツ・フレーム84内に示されるアイテム「MCOMMAND」が変更を必要とする場合、前記ユーザは、コンテンツ・フレーム84内のこのアイテムを選択し、前記ダイアログ110内の「はい」オプションを選択し、その後で、前記プロセスの現在のバージョンを格納しているコンフィギュレーション・データベース100がこのアイテムに関連付けられたデータを取り出すためにアクセスされる。取り出されたデータは、テキスト形式又はグラフィック形式の如き一又は複数の方法で表示され得る情報を表わす。このように、前記コンフィギュレーション・アプリケーション96（例えば、Control Studio（商標））によって提供される機能性には、取り出された情報を表示又は変更することが必要であり得る。何れにしても、適切なアプリケーションが、前記情報を見るために、また、任意の好ましい変更をなすために利用され、その時には、前記ユーザは、主として保存タスクを実行する。

【００３８】或るアイテムのコンフィギュレーションを見ることだけに関するコンフィギュレーション・アプリケーション96によって提供される機能性は、上記VCATシステム98との統合によって影響を受ける必要はないことを注記しておく。即ち、上記VCATシステム98は、ユーザに、チェックアウト／チェックイン手順を要求すること又は開始させることなく、前記コンフィギュレーション情報の「読み出しのみ」のコピーを見させることが好ましい。

【００３９】前記アイテムに関連付けられたこのような変更情報を格納するための保存タスクの開始は、前記アイテムのチェックインに好ましくは先行する。詳述すれば、前記保存タスクの実行によって、まず、コンフィギュレーション・アプリケーション96がコンフィギュレー

ション・データベース100において前記アイテムに関連付けられた変更情報を表わすデータを格納するようになる。次に、上記VCATシステム98は、バージョン制御データベース102において、変更情報を表わすデータを格納するためにチェックイン・タスクの実行を開始する。該チェックイン・タスクの実行においては、ユーザにチェックイン操作に関してコメントを入力させるための別のダイアログ（図示せず）を生成することを含むことが可能である。このコメントは、例えば、前記コンフィギュレーションへの変更の背景となる理由を指し示すことが可能である。前記コメントを表わすデータは、それから、バージョン制御データベース102に格納され、変更されたコンフィギュレーションを表わすデータに関連付けられる。

【００４０】チェックアウト／チェックイン環境は、コンフィギュレーション・データベース100においてバージョナブル（versionable）であるアイテムだけに限られていることが好ましい。一般に、バージョナブル・アイテムは、履歴のコンフィギュレーション情報が該アイテム自体に関連して維持又は格納される任意のアイテムを含むと理解されるべきである。好ましい実施形態において、バージョナブル・アイテムは、任意のモジュール，複合機能ブロック等のような、そのコンフィギュレーションが設計されているアイテムを含んでいる。その他のアイテム（例えば、モジュール・パラメータ）は、バージョナブルであると見なされないが、履歴情報は、モジュール・パラメータ又はその他の非バージョナブル（non-versionable）・アイテムを含む前記アイテムのために格納されたコンフィギュレーション情報を通じて格納されることが可能である。非バージョナブル・アイテムの別の例としては、ステップ又は移行がある。

【００４１】１つの実施形態において、前記チェックアウト操作は、単一のチェックアウトだけができる程度に限られている。このように、２人以上のユーザが、同一アイテムをチェックアウトしようとすることはあり得ない。ユーザが以前チェックアウトされたアイテムをチェックアウトしようとするイベントにおいて、上記VCATシステム98は、ユーザに、前のチェックアウトに責任があるユーザのアイデンティティーとともに、前のチェックアウトを知らせるために、ダイアログ・ウィンドウ（図示せず）を生成する。これに代えて、上記VCATシステム98は、各アイテムの並列変更を追跡するために必要な機能性を有することが可能である。

【００４２】別の実施形態において、前記チェックアウト操作の開始は、上記VCATシステム98に、バージョン制御データベース102内のアイテムが、選択されたアイテムへの変更によって影響を受けることがあり得るかについて判定させるべく、前記選択されたアイテムを解析させる。その他のアイテムは、例えば、これらが前記選択されたアイテムを利用する場合に影響を受けることがあり得る。前記選択されたアイテムへの変更によって影響を受けることがあり得る前記アイテムは、それから、チェックアウトされたアイテム（もしあれば）もユーザが選択するできるようにするために、上記VCATシステム98によって生成されたダイアログ・ウィンドウ（図示せず）において確認されることができる。同様のダイアログ・ウィンドウは、前記選択されたアイテムによって参照され、依存され、又は前記選択されたアイテムに関連付けられた任意のチェックアウトされたアイテムについて、ユーザに警告するためのチェックイン手順の間、生成されることがあり得る。

【００４３】前記チェックアウト及びチェックイン操作を開始するオプション、又は上記VCATシステム98内にで実行されるその他のどの様なタスクも、多くの方法でユーザ・インタフェース94内に提供されることが可能である。前記コンフィギュレーション・アプリケーション96に関連して上述され、また、図示されたようなウィンドウ型の環境においては、ユーザが、最上段のメニュー・バー112から「ツール」オプションを選択することがあり得、これによって、公知のウィンドウズ手法に従って複数の利用可能なタスクを表示させる。同一又は同様のメニューは、コンフィギュレーション・アプリケーション96のメイン制御ウィンドウ80において、或るアイテム上で「右クリック」することによって、ユーザ・インタフェース内に生成され得る。この場合、前記メニューは「状況メニュー」と称され得る。

【００４４】図６及び図７は、上記VCATシステム98によってユーザ・インタフェース94内に生成され得るドロップダウン・メニュー114及び状況メニュー116の一例をそれぞれ示している。前記ドロップダウン・メニュー114は、ポインティング・デバイス等を介してユーザによって選択され得る複数のタスク・アイテムを有している。バージョン制御アイテム118は、公知のウィンドウズ手法に従ってバージョン制御サブメニュー120を生成すべく、ハイライトされて（ユーザによる適切な選択の後で）示される。順番に、バージョン制御サブメニュー120は、上記VCATシステム98によって開始及び／又は実行され得るタスクに応じた複数のタスク・アイテムを有している。前記状況メニュー116は、メイン制御ウィンドウ80内で選択されたアイテムに基づいて動作することができるVCATシステム98のタスクに応じた複数のタスク・アイテムを同様に有している。どちらのメニューにおいても、如何なる理由であっても実行が不能（例えば、不適合又は無許可）となっているコマンドは、異なるフォント，スタイル等で無効であるものとして表示されることが可能である。例えば、図６のサブ・メニュー120において、恐らく、選択されたアイテムは未だチェックアウトされておらず、従って、前記チェックイン・タスクは開始することができない。前記メニュー116，120を通じて利用できるようになされたタスクの各々は、上記VC

ATシステム98の操作に関連して以下に説明される。

【００４５】さて、図８を参照して、ユーザが、前記状況メニュー116（図７参照）において「オプション」アイテムを、又は、これに代えて、別のメニュー（図示せず）に記述された「優先」アイテムを選択した場合、バージョン制御オプション・ダイアログ122が、上記VCATシステム98によって生成され、ユーザ・インタフェース94で表示される。バージョン制御オプション・ダイアログ122は、ユーザに、上記VCATシステム98によって生成された環境に対するユーザ選択を実現することを可能にする。下記の表１で詳細に記述されるように、前記バージョン制御オプション・ダイアログ122には、３つの対応するオプション間で切り換えるチェックボックスが設けられている。

【００４６】

【表１】

バージョン制御オプション・ダイアログ

名称	種類	最小	最大	デフォルト	内容
手動チェックアウト	ラジオボタン	該当なし	該当なし	選択	変更のためにアイテムのチェックアウトをユーザに促すべくダイアログを表示するか否かを定める。
自動チェックアウト	ラジオボタン	該当なし	該当なし	非選択	アイテムが自動的にチェックされるか否かを示す。
自動チェックアウト及び自動チェックイン	ラジオボタン	該当なし	該当なし	非選択	アイテムのコンフィギュレーションを変更するときに、該アイテムを自動的にチェックアウト及びチェックインするか否かを示す。

【００４７】第１のオプションでは、ユーザが手動でアイテムをチェックアウトすることを望むか否か定める。前記バージョン制御オプション・ダイアログ122中の第２のオプションは、ユーザがアイテムの変更をしようとしたとき、又はアイテムの変更を開始したときに、該アイテムが自動的にチェックアウトされるどうかについてのものである。アイテムが自動的にチェックアウトされる場合には、上記VCATシステム98は、コンフィギュレーション・アプリケーション96を利用しているプロセス設計者にとって判り易いものとなる。第３及び最後のオプションは、ユーザが変更のための機会を用意するセッションの後でアイテムを保存しようとするときに、アイテムが、自動的にチェックアウト及びチェックインされるか否かについてのものである。

【００４８】前記バージョン制御オプション・ダイアログ122は、各ユーザによって選択されるべき又は有効とされるべきその他のオプションを含むことが可能であることを注記しておく。例えば、各ユーザは、チェックイン操作の間、コメントを提供しないオプションを与えられることがあり得る。しかし、このようなコメントの供給に関する更なる詳細は、表２に関連して下記に提供される。

【００４９】ユーザは、自動チェックアウトを選択することを望むかも知れない。なぜなら、例えば、バージョナブル・アイテムへの変更が、一又は複数のその他のバージョナブル・アイテムに影響を及ぼし、また、該一又は複数のその他のバージョナブル・アイテムへの変更に起因し得るからである。例えば、複合機能ブロックのようなアイテムの変更は、該複合機能ブロックを使用する一又は複数のモジュールに影響を及ぼすことがある。上記VCATシステム98は、各チェックアウト操作の間、その他のバージョナブル・アイテムが或るアイテムのコンフィギュレーションを変更するためにチェックアウトされる必要があるかについて決定することが好ましい。これらのその他のバージョナブル・アイテムの変更は、「必然的な変更」と称されることがある。ユーザが手動チェックアウトを選択した場合、上記VCATシステム98は、ユーザにこれらのアイテムをチェックアウトさせる。自動チェックアウトが有効とされる場合、上記VCATシステム98は、ユーザに促すことなく、前記必然的な変更が発生し得るその他のアイテムの各々を自動的にチェックアウトする。

【００５０】ユーザは、バージョン制御サブメニュー120又は状況メニュー116を通じて提供される適切なコマンドを利用することによって、アイテムを手動でチェックアウト及びチェックインすることが可能である。好ましくは、上記VCATシステム98は、それから、選択されたアイテムがバージョナブル・アイテムであるか否かを判断する。この時、上記VCATシステム98は、前記選択されたアイテムが既にチェックアウトされているか否かも判断する。前記選択されたアイテムが既にチェックアウトされている場合には、メッセージ・ダイアログ（図示せず）は、この事実についてユーザに警告し、例えばボタンを通じて、前記メッセージ・ダイアログを閉じ、ユーザ・インタフェース94の前の状態に戻るための機会をユーザに提供する。

【００５１】前記プロセス10のコンフィギュレーションが階層的な方法において記述されるので、上記VCATシステム98は、バージョナブルである下位アイテムを有するアイテムのチェックアウトを許容させなければならない。従って、チェックアウト操作の間、ユーザは、このような任意のアイテムを繰り返しチェックアウトするオ

プションを与えられている。同様のオプションは、チェックイン操作に関しても与えられる。１つの実施形態において、チェックアウト又はチェックインの繰り返しがユーザによって選択される場合、上記VCATシステム98は、チェックアウト（又はチェックイン）され得るバージョナブルな下位アイテムのリストをユーザに提供するダイアログ・ウィンドウ（図示せず）を生成する。それから、ユーザは、前記リスト化されたアイテムの何れか１つを選択（又は選択解除）し、選択的な繰り返し操作を開始する。

【００５２】上記VCATシステム98がコンフィギュレーション・アプリケーション96に統合されるとき、前記メイン制御ウィンドウ80内に表示されたアイテムの外見は、該アイテムがチェックアウトされていることを意味すべく変更されることが可能である。１つの実施形態において、チェックマークは、前記アイテムに関連付けられたアイコンを覆う。その他の様々な表現方法が利用され得ると理解されるべきである。異なる色のチェックマークを、特定のユーザを示すために、又はアイテムが現在のユーザ又は異なるユーザによってチェックアウトされた＊ときに利用することも可能である。

【００５３】さて、図９を参照して、チェックアウトされたアイテムは、前記バージョン制御サブメニュー120及び状況メニュー116において利用可能とされた適切なコマンドを使用して手動でチェックインされることが可能である。ユーザが手動チェックイン環境を続行することに決めたとき、チェックイン・ダイアログ・ウィンドウ124は、前記チェックイン手順のユーザの開始に基づいて生成されることが好ましい。前記チェックイン・ダイアログ・ウィンドウ124は、前記アイテムの変更に関してユーザが供給したコメントを受け付けるためのコメント・フィールド126を有している。コメントは、例えば、変更がなされた理由の説明を含むことが可能である。前記コメント・フィールド126に加えて、チェックイン・ダイアログ・ウィンドウ124は、下記の表２で詳細に記述されるその他の幾つかのオプションをユーザに提供する。

【００５４】

【表２】

バージョン制御チェックイン・ダイアログ

名称	種類	最小	最大	デフォルト	内容
繰り返し	チェックボックス	該当なし	該当なし	チェックしない	このユーザによってチェックされた任意の下位アイテムにチェックインし、コンフィギュレーション階層の或るレベルにだけ好ましくは適用可能である。
チェックアウトを継続	チェックボックス	該当なし	該当なし	チェックしない	VCATシステムがアイテムをチェックアウトし続けるか否かを示す。
違い	ボタン	該当なし	該当なし	該当なし	コンフィギュレーション・データベース内の（チェックインされるべき）最新バージョンと現在のバージョンとの違い。
コメント	編集フィールド	該当なし	該当なし	空白	変更についてユーザが供給した説明。

【００５５】特に、ユーザには、チェックインされるべきアイテム（これはチェックアウトもされるべき）に関連付けられた任意の下位アイテムがチェックインされるべきであることを示すオプション、また、アイテムが更なる変更のためにチェックアウトされ続けるべきであることを選択するオプションが付与されている。上述のように、前記チェックイン操作は、バージョン制御データベース102における前記コンフィギュレーションを表わすデータも、最新のコンフィギュレーション・バージョンとして格納する。しかし、この場合、たとえ前記アイテムがチェックインされているとしても、該アイテムのコンフィギュレーションへの更なる変更をなす機会は、アイテムがチェックアウトされたままであるので、存続し続ける。

【００５６】チェックイン・ダイアログ・ウィンドウ124は、前記チェックイン手順を実行又はキャンセルするための「ＯＫ」ボタン128及び「キャンセル」ボタン130をそれぞれ備えるとともに、前記アイテムの現在のバージョン（コンフィギュレーション・データベース100に格納されたデータによって表わされる）と、チェックインされるべき変更されたバージョンとの比較を開始するための「違い」ボタン132を備える。比較情報は、コンフィギュレーション・データベース100とバージョン制御データベース102内の最新のバージョンとにアクセスすることで上記VCATシステム98によって生成され、ユーザ・インタフェース94を介して提示される。「違い」情報の生成及び提示は、以下に更に詳細に説明される。

【００５７】或るアイテムにチェックインするにあたって、上記VCATシステム98は、前記アイテムがチェックインされていることを示すために該アイテムの外見を変更

することが可能である。この外見の変更は、例えば、それに関連するアイコン上のチェックマークの除去に相当することがあり得る。上記VCATシステム98は、チェックアウトされたアイテムがチェックマーク等によってこのように示されることを確実にするために、ユーザに対して、コンフィギュレーション・データベース100におけるアイテムの全てのための状態更新を要求することができるようにする機能性を更に有することが可能である。例えば、２人以上のユーザが同時に前記プロセスを設定することに取り組んでおり、前記ユーザのうちの１人が別のユーザのユーザ・インタフェース94上に示されたアイテムを最近チェックアウトした場合、このような機能性は必要であり得る。

【００５８】チェックイン・ダイアログ・ウィンドウ124内の「キャンセル」ボタンの選択は、チェックイン操作を開始することなくチェックイン・ダイアログ・ウィンドウ124を閉じる。

【００５９】しかし、上記VCATシステム98は、アイコン上のチェックマークを削除する「チェックアウトを元に戻す」タスクをユーザが開始することも可能とし、そして、前記アイテムが変更される場合、バージョン制御データベース102から最新のバージョンを示すデータを取り出し、該データをコンフィギュレーション・データベース100にインポートする。この手順は、チェックアウト操作の時点で現在のバージョンに、バージョナブル・アイテムのコンフィギュレーションを元に戻す。このタスクは、上記VCATシステム98に関連付けられたドロップダウン又は状況メニューを介して利用できるようにすることが可能である。

【００６０】前記コンフィギュレーション・アプリケーション96は、チェックアウト／チェックイン操作を開始する追加の方法を提供することが可能である。例えば、ユーザは、Explorerシステム内のアイテムを選択することが可能であり、また、右クリック又は同様のポインタ操作を通じてそれに関連付けられた「プロパティ」にアクセスすることが可能である。それから、プロパティ・ウィンドウは、前記ウィンドウ内に表示された様々な「プロパティ」フィールドの内容について編集することを提供する標準的なウィンドウズ手法に従って生成される。モジュールは、例えば、それに関連付けられたテキスト形式の記述を有することが可能であるか、又は前記プロセス内で実行中の或るパラメータを利用することが可能である。テキスト形式の記述及びパラメータの値は、前記モジュールの「プロパティ」として指定され得る。ユーザがプロパティを変更して「OK」又は「適用」ボタンを選択する場合、上記VCATシステム98は、上述したようにチェックイン手順が開始された後で、（自動チェックアウトが有効とされていなければ）前記アイテムをチェックアウトすることをユーザに促す。同様の動作は、プロセス・コンフィギュレーション・アプリケーションのコマンド・メニューにおいて利用可能とされた「開く」コマンドの利用を通じて生成され得る。

【００６１】ユーザが前記Explorerシステム内のアイテムの名前を変更しようとするとき、チェックアウト手順は、上記VCATシステム98によって開始されることも可能である。標準的なウィンドウズ手法に従って、ユーザは、アイテムに関連付けられた名前を、それを編集することを許容すべく、選択することが可能である。一旦、ユーザが名前を編集し終わり、変更に入ろうとすると、前記アイテムがチェックアウトされる場合に限って新しい名前が受け付けられる。アイテムが既にチェックアウトされている場合、前記アイテムは、コンフィギュレーション・データベース100において名前を変更され、前記アイテムの新しいバージョンを示すデータは、それがチェックインされるときに、バージョン制御データベース102に格納される。このとき、上記VCATシステム98は、名前を変更する操作を記述するコメントを表わすデータを更に生成することが可能であり、それから、このようなデータを、前記アイテムの新しいバージョンを示すデータと関連付けることが可能である。名前を変更されるべきアイテムが未だチェックアウトされていなければ、前記アイテムを、ダイアログ・ウィンドウ（図示せず）を介してチェックアウトする趣旨を（自動チェックアウトが有効とされていなければ）、ユーザに促す。

【００６２】上述のように、アイテムの各々の前のコンフィギュレーションを表わすデータは、それに割り当てられたバージョンを反映するデータとともに、バージョン制御データベース102に格納される。前記バージョンは、数字によって識別されることが好ましいが、その他の任意の方法で示されることが可能である。バージョン番号に関連付けられた各々の前のコンフィギュレーション有することは、前記プロセスがランタイム環境であるときと同様に、バージョン制御データベース102の内容の解析の間、ユーザを補助することが可能である。このために、アイテムがプロセス・コントローラ12及び／又はバッチ・エグゼクティブ・ルーチンにダウンロードされるときに、前記アイテムのバージョン番号又は識別子が、アイテム・パラメータとして含まれていることが好ましい。（前記バッチ・エグゼクティブ・ルーチンは、バッチ・プロセスの実行を管理し、これに接続するプロセス・コントローラ12を監視するためにホスト・ワークステーション14上に常駐する。）換言すれば、プロセス制御システム10がプロセスを実行する準備をしているときにホスト・ワークステーション14からダウンロードされるデータは、ダウンロード操作に関係しているコンフィギュレーション・データベース100における或るアイテムのためのデータ（即ち、各モジュール，フェーズ，手続き的な要素等）を識別するバージョンを含んでいる。それから、前記バージョン情報は、ランタイム環境において、前記プロセス・コントローラ12及び／又はバ

ッチ・エグゼクティブ・ルーチンを通じてプロセス・オペレータによって利用可能である。前記バージョン情報についての知識は、コンフィギュレーション・アプリケーション96内で作業しているプロセス・オペレータとプロセス設計者との間の通信を補助することが可能である。

【００６３】アイテムをプロセス・コントローラ12にダウンロードすること、又はランタイム環境一般は、ダウンロード手順を通じてユーザを案内するユーザ・インタフェース94のためのコンフィギュレーション・アプリケーション96によって生成された一又は複数のダイアログ・ウィンドウ（図示せず）を含むことを注記しておく。これらのダイアログ・ウィンドウは、上記VCATシステム98がそれに統合されないとき、コンフィギュレーション・アプリケーション96によって生成された同一のウィンドウに対応することが可能である。しかし、上記VCATシステム98は、ダウンロードされるべき全てのアイテムがダウンロード手順を進められる前にチェックアウトされていないことを確かめることが可能である。もし何れかのアイテムがチェックアウトされた場合、上記VCATシステム98は、ユーザに状況を警告し、且つチェックアウトされたアイテムを識別するための一又は複数のエラー・ダイアログ・ウィンドウ（図示せず）を生成することが可能である。このようなエラー・ダイアログ・ウィンドウは、前記ダイアログ・ウィンドウにおいて識別された一又は複数のチェックアウトされたアイテムのチェックイン手順を開始させるように案内するために、その中にボタンを設けることによってチェックイン手順を容易にすることが可能である。

【００６４】さて、図１０を参照して、バージョン識別情報は、本願発明の１つの実施形態に係る上記VCATシステム98によって生成されたバージョン監査証跡レポート（以下、単に「監査証跡レポート」と称す）を通じてプロセス設計者が利用可能なようになされることが可能である。一般に、前記プロセス・コンフィギュレーションへの各変更又は修正は、前記プロセスに関連付けられた特定のバージョンとしてバージョン制御データベース102に記録される。前記バージョン及び前記変更の要旨を示すデータは、チェックインの日付及び時間，並びに前記アイテムにチェックインしたユーザ，及び前記変更の理由を表わすデータとリンクされることが可能ある。上記VCATシステム98は、また、チェックアウトの日付及び時間を表わすデータとともに、前記アイテムをチェックアウトしているユーザのアイデンティティーも格納することが可能であることを注記しておく。

【００６５】前記監査証跡レポートは、ユーザ・インタフェース94に対して上記VCATシステム98によって生成された監査証跡ダイアログ・ウィンドウ140を介してユーザに提供されることが可能である。詳述すれば、「履歴表示」タスクは、ユーザがコンフィギュレーション・アプリケーション96のうちの１つ内にある間、例えば、状況メニュー116を含む上述のテクニックのうちの１つを使用することによって前記ユーザによって開始されることが可能である。履歴の監査証跡は、その他の任意の提示装置（例えば、プリンタ）介してユーザに提供されることも可能であり、又は、例えば、監査証跡を表わすデータがネットワーク上を電気的に伝送されるように、前記レポートの電子版として具現化されること可能である。

【００６６】前記監査証跡ダイアログ・ウィンドウ140は、バージョン番号，チェックアウトに関連付けられたユーザ名，チェックインの日付及び時間，並びに変更又は動作についての記述に関するバージョン情報を提示するための複数のフィールド名を識別するヘッダ144を有するテーブル部142を備えている。前記バージョン情報は、図１０に示す前記フィールド名の下方に行（又はレコード）として提示されている。各レコードは、ポインタを介して選択可能であり、様々な操作を開始することができる。前記監査証跡ダイアログ・ウィンドウに関連して、上記VCATシステム98によって提供される機能性は、下記の表３に要約してある。

【００６７】

【表３】

監査証跡ダイアログ・ウィンドウ

名称	種類	最小	最大	デフォルト	内容
バージョン，ユーザ，日付，動作	リスト制御	該当なし	該当なし	該当なし	各行はアイテムの履歴記録に対応する。
閉じる	ボタン	該当なし	該当なし	有効	ダイアログを閉じる。
ロールバック	ボタン	該当なし	該当なし	有効	選択されたバージョンにロールバックする。
違い	ボタン	該当なし	該当なし	有効	２つの選択されたバージョン又は選択されたバージョンを現在のバージョンと比較する。
詳細	ボタン	該当なし	該当なし	有効	選択されたバージョンのコメントを表示する。
表示	ボタン	該当なし	該当なし	有効	エキスポート可能なアイテムをテキスト形式又はグラフィカル形式で表示する。

【００６８】表３に記述したように、ユーザは、そのバージョンに対応するテーブル部142の行を選択し、それから、表示ボタン143を選択することによって、前記アイテムのバージョンの詳細を見ることが可能である。それから、前記バージョン制御データベース102に格納されたデータは、アイテムの性質に基づいて、テキスト形式又はグラフィカル形式の何れか又は両方で前記アイテムの選択されたバージョンのコンフィギュレーション情報を提示すべくアクセスされる。或るアイテムは、例えば、テキストでのみ構成されることがあり得、従って、グラフィカル形式で見られることがないかも知れない。例えば、ワークステーション又はI/Oデバイスのようなバージョナブル・アイテムは、変更されることができるパラメータのみを含んでいる。前記パラメータは、テキストで記述され、従って、このようなアイテムにはグラフィカル形式の表示は利用できない。しかし、或るアイテムは、グラフィカル形式及びテキスト形式で見られることがあり得、アイテム（例えば、モジュール，複合機能ブロック，又は一連のフローチャート・アルゴリズムに基づいたその他のアイテム）を一般に有することが可能である。そのような場合に、ユーザは、前記フォーマット（形式）のうちの１つを選択することを促され得、又は、これに代えて、両方のフォーマットが提示されることがあり得る。

【００６９】テキスト形式で示されるアイテムの一例は、図１１に示されており、これはユーザ・インタフェース94のために上記VCATシステム98によって生成された画面表示である。該画面表示は、アイテムに関連付けられたテキスト形式の情報のためのフレーム148を有するテキスト表示ダイアログ・ウィンドウ146を備える。該テキスト表示ダイアログ・ウィンドウ146は、ユーザに、以下に説明する幾つかの表示オプション又は操作を提供する。

【００７０】図１２は、グラフィカル形式で示されるアイテムの画面表示の一例である。テキスト形式用の画面表示のように、グラフィカル形式は、アイテムに関連付けられたグラフィカル形式の情報のためのフレーム152を有するグラフィカル形式の表示ダイアログ・ウィンドウ150を通じて提示される。

【００７１】図１１及び表３に戻って、ユーザは、また、前記要求されたバージョンを強調表示させた後に、詳細ボタン154を選択することによって、特定のバージョンのために、チェックインの時点で記録される如何なるコメントも解析することが可能である。前記コメントは、チェックイン・ダイアログ・ウィンドウ124を通じて、好ましくは格納される。しかし、その他の方法は、特定のバージョンとともにコメントを表わすデータを関連付けるために利用されることがあり得る。前記監査証跡ダイアログ・ウィンドウ140は、さらに、ユーザに、ポインタ（又はその他の選択機構）を使用することによって、アイテムの２つのバージョンをテーブル部142に対応する２つの記録を強調表示させるように選択させる。２つの記録を強調表示させることは、ユーザに、違いボタン156を選択することによって違い操作を開始させることを可能にし、それによって、上記VCATシステム98に、ユーザが前記２つのバージョンを比較することができるようにするフォーマットで、アイテムの各々の選択されたバージョンのためにコンフィギュレーション情報を提供する違いウィンドウ・ダイアログを生成させる。前記違い操作については、以下に更に詳細に説明する。単一の記録がハイライトされた場合、及び違いボタン156が選択された場合には、上記VCATシステム98は、前記選択されたバージョンが、コンフィギュレーション・データベース100に格納されたアイテムの現在のバージョンと比較されるように、前記違いウィンドウを生成する。

【００７２】前記監査証跡ダイアログ・ウィンドウ140は、通常、ユーザが、アイテムのコンフィギュレーションの前のバージョンのうちの１つを実行することも可能にする。詳述すれば、上記VCATシステム98は、前記選択されたアイテムに関連付けられたコンフィギュレーション・データベース100に格納されたデータを変更するた

めに、前のコンフィギュレーション・バージョンを表わすデータを取り出すべくバージョン制御データベース102にアクセスするルーチンを開始することを、ユーザによって指示されることが可能である。それから、コンフィギュレーション・データベース100の変更は、コンフィギュレーション・データベース100にデータをバージョン制御データベース102からインポートすることによって発生することがあり得る。アイテムの前のコンフィギュレーション・バージョンに、及び１つの実施形態に従って、この「ロールバック」を実行するために、監査証跡ダイアログ・ウィンドウ140は、テーブル部142での記録が一旦選択されてから、ユーザによって選択されることがあり得るロールバック・ボタン158を有している。

【００７３】１つの実施形態において、上記VCATシステム98は、ランタイム環境での使用のためにプロセス・コントローラ12にダウンロードされたコンフィギュレーションに、ロールバックするためのオプションをユーザに提供する。このようなロールバック操作は、ドロップダウン又は状況メニューから「ダウンロードを回復」の如きタスク・アイテムの選択によって開始されることが可能であり、ダウンロードされたそれらのバージョンをリスト化するだけである特別の監査証跡ダイアログ・ウィンドウ（図示せず）の、上記VCATシステム98による生成を含む。以下に更に詳細に説明するように、バージョン制御データベース102は、どのバージョンがランタイム環境での使用ためにダウンロードされたかを示すデータを有することが好ましい。それから、バージョンは、ユーザによって選択されることがあり得る。そして、そのどちらかのコンフィギュレーション情報が、上記VCATシステム98によってバージョン制御データベース102から取り出され、最新のコンフィギュレーション・バージョンとして格納される。

【００７４】上記VCATシステム98は、コンフィギュレーション・データのインポート操作がコンフィギュレーション階層内のその他のアイテム及び該アイテムに依存するプロセスについてチェックすることに成功したか否かを判定することが好ましい。例えば、モジュールのコンフィギュレーション・バージョンが回復しているとき、該コンフィギュレーション・バージョンは、或る下位アイテム（例えば、複合機能）ブロックが存在することを要求することがあり得る。それから、前記モジュールは、前記ロールバック操作の成功がその下位アイテムの継続的な存在に依存する観点において「依存性」を有すると言える。モジュールは、依存性（例えば、埋め込まれた機能ブロック，結合された機能ブロック，エミュレーション・セット，アラームの種類，その他のモジュール，プラント・エリア，及びコントローラ）の多数の異なる種類を有することが可能である。従って、１つの実施形態において、上記VCATシステム98は、任意のバージョナブル・アイテムが削除されているかを判定するために、ロールバックの対象となっているアイテムの各々の下位アイテムをチェックする。

【００７５】好ましい実施形態において、前記コンフィギュレーション・データベースにインポートされたデータは、アイテムの最新のコンフィギュレーション・バージョンとともに、バージョン制御データベース102にチェックインされる。該チェックインは、コンフィギュレーション・データベース100の変更と並行して発生し、ユーザがアイテムの前のコンフィギュレーション・バージョンへ戻ったことを示すコメントを表わすデータのストレージを含むことが好ましい。これに代えて、上記VCATシステム98は、カスタマイズされたコメントを入力する、デフォルトのコメント入力する、又は全くコメントを入力しないオプションをユーザに提供すべく、ダイアログ・ウィンドウを生成することが可能である。前記ロールバック・タスクが開始される前にアイテムがチェックされた場合には、上記VCATシステム98は、前記ロールバック・タスクが完了された後でアイテムをチェックアウトし続ける。

【００７６】最後に、監査証跡ダイアログ・ウィンドウ140は、監査証跡ダイアログ・ウィンドウ140を閉じて、その前の状態にユーザ・インタフェース94を戻すためのボタン160を有している。

【００７７】図１０に示すように、前記監査証跡レポートにおいて記録される動作（チェックイン操作以外）のうちの１つは、プロセス・コンフィギュレーション全体にラベルを設定する操作である。さて、図１３を参照して、ユーザは、例えば、コンフィギュレーション・アプリケーションを使用して設計されたモジュールのうちの１つ内のバージョナブル・アイテムを選択することによって、上記VCATシステム98によって生成されたドロップダウン又は状況メニューの何れかを介してラベルセット・タスクを選択することによって、ラベルセット操作を開始することが可能である。これに応じて、上記VCATシステム98は、ラベル・フィールド164及びコメント・フィールド166を有するラベルセット・ダイアログ・ウィンドウ162を生成する。それから、バージョン制御データベース102における前記選択されたアイテム（及び下位アイテム）に適用するラベルの入力を、ユーザに促す。それから、バージョン制御データベース102に格納された各アイテムの最新のコンフィギュレーション・バージョンには、ユーザが入力したラベルが割り当てられる。また、バージョン制御データベース102に格納された監査証跡データは、前記ラベルの割り当てを反映すべく更新されるが、最新のコンフィギュレーション・バージョンに適用されることが好ましい。詳述すれば、監査証跡のコンフィギュレーション履歴データ内のバージョン動作は、前記バージョンが入力された名前を有する新しく割り当てられたラベルに対応するように反映し、こ

れに関連付けられた詳細は、ラベルセット・ダイアログ・ウィンドウ162のコメント・フィールド166にユーザによって入力された情報に対応する。

【００７８】ラベルをセットすることは、様々なアイテムのコンフィギュレーション・バージョンがプロセスでの実施のためにプロセス・コントローラ12にダウンロードされた場合、プロセス設計者が「コントローラ1にダウンロードされた」の如きラベルによって手動で注記することを可能とするように、プロセス設計者にとっては利点がある。しかし、上記VCATシステム98は、前記ランタイム・デバイスにダウンロードされているように、コンフィギュレーション・バージョンにフラッグを立てるべく、各々のダウンロードされたアイテムに対してラベルを自動的に割り当てることが好ましい。ダウンロードの日付及び時間、並びに並列ダウンロードされたアイテムのような更なる情報を表わすデータを、前記ラベルに関連して格納することも可能である。

【００７９】従って、上記VCATシステム98は、プロセス全体又はその幾つかの部分のコンフィギュレーションを、バージョン制御データベース102において影響を受けたアイテムの各々に対して上述の手順でセットされたラベルに関連付けられたバージョンに回復する機能を提供することが可能である。１つの実施形態において、この機能は、コンフィギュレーション・アプリケーション96を使用して展開されたコンフィギュレーション階層におけるルート（又は最上位層の）アイテムのみに適用される。このような事例においては、「ラベルにシステムを回復する」タスクは、ユーザが、ルート・アイテムを選択し、ドロップダウン・メニュー又は図7に示されたメニューと同様の状況メニューから適切なタスク・アイテムを選択しようとするときに、開始され得る。或るラベルに前記システムを回復する意図をユーザに確認するために、上記VCATシステム98が、警告ダイアログ（図示せず）を生成することも可能である。それから、「ラベルにシステムを回復する」タスクが望ましいとユーザが示した場合には、当該システムが回復された前記ラベルの名前をユーザに促すことも可能である。これに代えて、上記VCATシステム98は、前記システムを、バージョン制御データベース102にセットされた最後のラベルへ自動的に回復することも可能である。何れにしても、システムをラベルに回復するタスクを一旦実行すると、上記VCATシステム98は、各アイテムに対するラベルに関連付けられたコンフィギュレーション・バージョンを表わすデータを取り出すべく、バージョン制御データベース102にアクセスし、取り出したデータをコンフィギュレーション・データベース100にインポートする。それから、上記VCATシステム98は、各アイテムに対する新しいコンフィギュレーション・バージョンを示すデータを、前記ラベルへの復元を反映するそれに関連付けられたデータとともに格納することによって、復元を反映すべく、バージョン制御データベース102を変更する。１つの実施形態において、このデータは、前記ラベルへの復元を示すコメントを表わすデータを格納することによって、コンフィギュレーション・バージョンに関連付けることが可能である。

【００８０】１つの実施形態によれば、上記VCATシステム98は、コンフィギュレーション・データベース100から削除又は除去されたアイテムを表わすデータを、バージョン制御データベース102に依然として格納することが好ましい。このような場合には、バージョン制御データベース102は、前記削除されたアイテムを利用してプロセス・コンフィギュレーションに回復することをユーザが決めることに依存する。さて、図１４を参照して、上記VCATシステム98は、ユーザに、コンフィギュレーション・データベース100から削除されているアイテムを除去できるようにすることも可能である。そのために、上記VCATシステム98は、このような削除されたアイテムのリストを有する回復／除去ダイアログ・ウィンドウ168を生成する。該回復／除去ダイアログ・ウィンドウ168は、ドロップダウン又は状況メニューにおける回復／除去タスク・アイテムの選択に基づいて生成されることも可能である。このようなタスク・アイテムは、コンフィギュレーション・アプリケーション96でのアイテムの選択の後で、使用可能又はユーザが利用できるようにすることも可能である。前記選択されたアイテムがコンフィギュレーション・データベース100から削除された一又は複数の下位アイテムを有する場合には、ユーザは、前記削除されたアイテムの一又は複数を表わすデータをコンフィギュレーション・データベース100に追加すべく、回復／除去ダイアログ・ウィンドウ168を介して回復操作を開始することが可能である。

【００８１】前記回復／除去ダイアログ・ウィンドウ168は、前記ユーザによって選択されているリストにおける一又は複数の削除されたアイテムを元に戻すための回復ボタン170を有している。削除されたアイテムを回復するときに、上記VCATシステム98は、前記削除されたアイテムの最新のコンフィギュレーション・バージョンを取り出すべく、バージョン制御データベース102にアクセスし、それから、これに関連付けられたデータをコンフィギュレーション・データベース100にインポートすることが可能である。これに代えて、上記VCATシステム98は、バージョン制御データベース102に格納されている複数のバージョンの程度、以前のコンフィギュレーション・バージョンの選択をユーザに促すことも可能である。

【００８２】一方、回復／除去ダイアログ・ウィンドウ168において提供される除去ボタン172の選択は、バージョン制御データベース102から前記アイテムに関連付けられた任意のデータを除去する結果となる。前記回復／除去ダイアログ・ウィンドウ168に関する更なる詳細

は、下記の表４で提供される。

【００８３】

＊【表４】

＊

回復／除去ダイアログ・ウィンドウ

名称	種類	最小	最大	デフォルト	内容
アイテム	チェックリストボックス	該当なし	該当なし	選択された親に基づいて削除された下位アイテムのリスト	選択された親に基づいて削除された全ての下位アイテムをリスト化する。
回復	ボタン	該当なし	該当なし	該当なし	データをバージョン制御データベースから取り出し、コンフィギュレーション・データベースにアイテムをインポートすることによって、アイテムをコンフィギュレーション・データベースに戻す。
除去	ボタン	該当なし	該当なし	該当なし	アイテム及び全ての監査証跡履歴データは、永久的にバージョン制御データベースから削除される。

【００８４】本願発明の１つの実施形態によれば、上記VCATシステム98は、コンフィギュレーション・アプリケーション96によって生成されず、展開されず、又は管理されないアイテムに対するバージョン制御及び監査証跡の機能性をサポートする。例えば、ユーザは、特定のアイテムの機能性を記述する文書処理プログラムで文書を生成することが可能である。コンフィギュレーション、即ち記述が変更されるときに、前記文書の如何なる編集であっても格納及び追跡することが望ましい。このために、Explorerシステムによってユーザ・インタフェース94に生成されたメイン制御ウィンドウ80は、新しいバージョン制御アイテムを指定すべく利用される。詳述すれば、第１のユーザは、「ユーザの作業領域」オブジェクト（例えば、Explorerシステムにおけるオブジェクト）を拡張する。それから、階層フレーム82内で新しいフォルダを生成するように案内されたタスク・アイテムは、ユーザによって選択され、新しいフォルダが、公知のウィンドウズ手法に従って生成されて名前を付けられる。同様に、新しいアイテムを生成するように導かれたタスク・アイテムが選択され、それから、ユーザは、最近生成されたフォルダ内に新しいアイテムとして挿入するための文書又はファイルに対してブラウズする機会を提供される。前記文書又はファイルは、前記フォルダに一旦挿入されると、バージョン及びバージョンに関連する情報が格納され得る別のアイテムとしてバージョン制御データベース102に追加される。その後、前記ファイル又は文書のバージョンの何れかが、ドロップダウン又は状況メニューを介してユーザが利用可能なようになしてある入手タスク・アイテムを使用してバージョン制御データベース102から取り出されることが可能である。前記入手タスクは、取り出されたファイル（例えば、フレキシブル・ディスク）のための所望するストレージ位置を促すダイアログ・ウィンドウ（図示せず）をユーザに提供する。それから、前記ファイル又は文書（例えば、文書処理アプリケーション）を生成するために利用されるネイティブ・アプリケーションが、前記所望するストレージ位置から前記ファイル又は文書にアクセスすべく使用され得る。

【００８５】さて、図１５を参照して、上記VCATシステム98は、ユーザ・インタフェース94を介してユーザにバージョン制御フィードバック情報の表示のためのメッセージ・ダイアログ・ウィンドウ174を生成することが好ましい。該メッセージ・ダイアログ・ウィンドウ174は、各バージョン制御操作のテキスト形式の説明を、それが発生する際に、文書化のテキスト・ボックス176を有している。通常、メッセージ・ダイアログ・ウィンドウ174は、クリア・ボタン178がユーザによって最後に選択された時点から発生した操作の履歴に亘って、ユーザがスクロールできるようにしてある。また、中止ボタン180は、次の機会にユーザがバージョン制御操作を中止することができるように設けられている。例えば、ユーザがロールバック操作を開始した場合、上記VCATシステム98による操作の実行は、例えば、該操作の実行が既に完了されていない限り、前記中止ボタン180の選択をもって停止されることが可能である。前記メッセージ・ダイアログ・ウィンドウ174は、バージョン制御操作が或る時間発生しなかった場合、公知のウィンドウズ手法に従って自動的に最小化することが可能である。それから、前記メッセージ・ダイアログ・ウィンドウ174は、次のバージョン制御操作の開始の際に、ユーザ・インタフェース94上に自動的に再表示される。

【００８６】さて、図１１に基づいて、上記VCATシステム98がテキスト表示ダイアログ・ウィンドウ146を介してアイテムのコンフィギュレーションのテキスト形式の表示が、ユーザに提供される方法を説明する。上記VCATシステム98は、バージョン制御データベース102におけ

る各アイテムのコンフィギュレーション情報をテキスト形式で表示することができるが、これは必須ではない。上述したように、或るアイテムは、その性質上、グラフィカル形式で表示されることもあり得るコンフィギュレーション情報を有することも可能である。

【００８７】なお、或るアイテムのためのコンフィギュレーション情報が記述される前記フォーマットは、本願発明の実施には無関係である。しかし、上記VCATシステム98は、アイテムの１つのコンフィギュレーション・バージョン見るときと、２つのバージョンを比較するときとの両方で必要なユーザによる水平方向のスクロール量を最小化するテキスト形式のフォーマットを利用することが好ましい。さらに、キーワード及びラベルがオブジェクトの種類及びプロパティの如き属性を確認すべく利用することが好ましい。アイテムの種類を識別するラベルが利用され得る例としては、「パラメータ（PARAMETER）」，「機能ブロック（FUNCTION_BLOCK）」，及び「コントローラ（CONTROLLER）」が含まれる。プロパティ・ラベルの例としては、「名前（NAME）」及び「説明（DESCRIPTION）」がある。それから、前記キーワード及びラベルは、適当な言語に翻訳され得る。例えば、日本語バージョンのユーザ・インタフェース94では、日本語でキーワードを表示する。前記言語における条件付きの及びその他の論理的な記述を、テキスト形式での表示用に、読み易いフォーマットに変換することも可能である。

【００８８】１つの実施形態において、各アイテムのためのテキスト形式のフォーマットは、開始（BEGIN）及び終了（END）の如き区切り言語を含んでいる。例えば、図１１に示すように、或るモジュールのコンフィギュレーションのバージョン番号「13」は、以下の「BEGIN」及び「END」によって区切られるパラメータを有している。

```
BEGIN Parameter "ABS__PRESS__CF"
END Parameter "ABS__PRESS__CF"
```

別の実施形態において、前記アイテムの名前は、以下の例の如きラベルに関して識別されないかも知れない。

```
BEGIN FUNCTION__BLOCK
        NAME = "AI1"
        DEFINITION = "AI"
        DESCRIPTION = "Analog Input"
        LEFT = 200
        TOP = 200
        HEIGHT = 125

        WIDTH = 400
END FUNCTION__BLOCK
```

前記テキスト表示ダイアログ・ウィンドウ146は、検索ボタン200、下方へ検索ボタン202，及び上方へ検索ボタン204を有している。該ボタン200，202，及び204は、一般に、公知のウィンドウズ手法に従ってポインタ等によりそこで選択された文字列に基づいて、フレーム148内に記述されているテキスト形式の情報までユーザを案内する。このために、前記検索ボタン200を選択することによって検索操作を開始することは、上記VCATシステム98に、前記フレーム148でのユーザの案内を容易にする検索ダイアログ・ウィンドウ（図示せず）生成させ得る。それから、或る文字列は、フィールドに入力され得るが、この文字列の第１の例が、例えば、前記検索ダイアログ・ウィンドウにおける「ＯＫ」ボタン（図示せず）を選択することによって検索されることが可能である。その後、ユーザは、前記ボタン202，204を、下方又は上方の何れかへそれぞれ前記文字列を検索すべく使用することが可能である。テキスト表示ダイアログ・ウィンドウ146を介して提供されるこれらの操作及びその他に関する更なる情報について、下記の表５で説明する。

【００８９】

【表５】

テキスト表示ダイアログ・ウィンドウ

名称	種類	最小	最大	デフォルト	内容
検索	ボタン	該当なし	該当なし	有効	検索文字列をユーザに入力させる。
下方へ検索	ボタン	該当なし	該当なし	無効	検索文字列の次の例に進む。
上方へ検索	ボタン	該当なし	該当なし	無効	検索文字列の前の例に進む。

【００９０】この時点で、テキスト表示ダイアログ・ウィンドウ146において記述されたテキスト形式の情報がバージョン制御データベース102に格納されたそれを表わすデータから上記VCATシステム98によって生成される方法をより詳細に記述することは、有益である。前記バージョン制御データが格納される方法が本願発明の或る観点の実施に重大でないことは、まず理解されるべきである。しかし、本願発明の１つの実施形態において、前記バージョン制御データは、ファイル型のフォーマットでバージョン制御データベース102に格納されることが好ましい。その他の実施形態においては、前記バージョン制御データベースがオブジェクト指向データベースを備えるように、前記データは、オブジェクト指向的に格納される。

【００９１】前記バージョン制御データベース102は、本願出願人であるフィッシャー・ローズマウント・システムズ社から入手可能な前記DeltaV（商標）データベース管理アプリケーションを使用して好ましくは管理さ

れ、データベース100，102の両方を取り扱うために必要な範囲で変更される。これに代えて、Microsoft社のSQL Server（登録商標）Enterprise Managerをこの目的に利用することも可能である。

【００９２】コンフィギュレーション・バージョンを表わすテキスト形式の情報を生成するために、上記VCATシステム98は、テキスト形式又はグラフィカル形式のフォーマットの何れかに変換することができる方法で、妥当なデータをエクスポートすべくバージョン制御データベース102に主としてアクセスするルーチンを実行する。このために、チェックイン操作の間、上記VCATシステム98は、XML（拡張可能なマークアップ言語）の如きマークアップ言語に従ってファイルにおけるバージョン制御データを表わすテキスト型の表示を格納する。この時点で生成された前記XML文書に含まれるテキストを、バージョン制御データベース102に格納される単一の文字列に順番に並べることも可能である。詳述すれば、１つの実施形態において、各バージョナブル・アイテムは、各コンフィギュレーション・バージョンに対応するデータベースのレコードを有することが可能である。その事例において、各コンフィギュレーション・バージョンのレコードは、該コンフィギュレーション・バージョンに関連付けられたバージョン制御データを表す、それに格納されたXMLテキストの単一の文字列を有することに供するフィールドを備えている。これらのコンフィギュレーション・バージョンのレコードは、バージョン制御データベース102（この場合、リレーショナル・データベース）において、複数のテーブルのうちの１つのテーブルを構成することが好ましい。前記リレーショナル・データベースは、以下のものを格納するように案内するその他のテーブルを有することが可能である。（1）各バージョナブル・アイテムが削除されたかどうか、現在のバージョンの識別子、前記アイテムが現在チェックアウトされているどうか、及びもし該当するならば、どこへ、（2）各バージョナブル・アイテムのための監査証跡情報など。

【００９３】好ましい実施形態において、或るアイテムのコンフィギュレーションを完全に規定するために必要なコンフィギュレーション・データの全ては、各バージョンに対して別々に格納される。これに代えて、前記コンフィギュレーション・データは、バージョンの違いを規定するのみである方法で格納され得、その場合には、複数のデータベース・フィールドに関連付けられたデータは、特定のバージョンに対するXML文書を展開すべくアクセスされなければならない。

【００９４】従って、バージョン制御データベース102に後でアクセスするときに、簡単に操作することができ、また、解析することができる構造を持つ方法で、前＊記バージョン制御データが記述される。図１１の例で示されるコンフィギュレーション情報を展開するために、上記VCATシステム98は、読み易いテキスト形式で前記コンフィギュレーション情報を提供すべく、前記バージョン制御データのXML表示を変換する。XML形式化されたデータが処理される方法は、当業者には容易に理解されるべきであるが、次に更に詳細に記述される。

【００９５】本願の出願時の段階で、XMLは、当業者には周知のマークアップ言語であり、ワールドワイド・ウェブ・コンソーシム（www.w3.org）によってXML 1.0として標準化中である。その他のデータ方式が利用され得る一方で、多数のマークアップ言語の何れか１つの使用によって、前記アイテムのコンフィギュレーションのテキスト形式及びグラフィカル形式の表示の生成を容易にすることが好ましい。

【００９６】一般的に言って、XMLで記述される文書は、単一のルート（又は文書）要素から始まるノードと同様に、宣言，コメント，及び処理命令からなる論理的な構造を有する。例えば、バージョナブル・アイテムは、制御方法の一部を表すために、最初の又はルート「モジュール」要素を有することが可能である。このモジュール要素は、名前（NAME）と呼ばれる属性ノードを含むことがあり得、前記モジュールの名前を含むテキスト・ノードを有する。加えて、前記モジュールに、該モジュールを記述するための要素ノードを含ませることも可能である。このような要素は、例えば、「アルゴリズムの種類（ALGORITHM__TYPE」，「説明（DESCRIPTION）」，「実行速度（EXECUTION SPEED）」，及び「モジュールの種類（MODULE__TYPE）」を含むことが可能である。これらの要素の各々は、前記コンフィギュレーションのその特定の一部を代表する値を保持するテキスト形式のノードを有することが可能である。前記モジュール要素ノードは、追加の要素を含む要素ノードを更に有することが可能であり、従って階層を生成する。例えば、モジュールは、「ステップ（STEP）」を含むことが可能であり、該ステップは、一又は複数の「動作（ACTION）要素」を有することが可能である。該動作要素は、順番に、「名前（NAME）」，「説明（DESCRIPTION）」，「動作の種類（ACTION__TYPE）」等の如き要素を含んでいる。

【００９７】このように、前記XMLフォーマットは、HTML文書にあるものと同様のタグを補助として用いてバージョン制御データを記述する。しかし、XML文書において、前記タグは、上述したように、格納されているデータの種類にカスタマイズされることが可能である。例えば、テキスト形式の表現が上述の如くに提供された機能ブロックは、XMLフォーマットで下記の如く記述される。

```
<Function__Block>
    <Name>All</Name>
```

```
<Name>All</Name>
<Definition>AI</Definition>
<Description>Analog Input</Description>
<Left>200</Left>
<Top>200 Top>
<Height>125</Height>
<Width>400</Width>
</Function__Block>
```

XML文書に格納されたデータは、前記バージョン制御データに関連付けられた複数のノードを有するデータ・ツリー構造を作成すべく前記文書を解析するために提供されるオブジェクト・モデルに従ってアクセスされる。本願発明の実施には、XMLに適用される際に、文書オブジェクト・モデルの如き幾つかの異なるオブジェクト・モデルの何れかを利用することが可能である。

【００９８】このように、ユーザ・インタフェース94を介してテキスト表示ダイアログ・ウィンドウ146で提供されるテキスト形式の情報は、コンフィギュレーション・バージョンに対する上述のオブジェクト・ノードを生成すべく、データベース・フィールドにXMLテキストをロードすることによって生成される。それから、前記文書オブジェクト・モデルは、コンフィギュレーション情報を引き出す目的から、また、選択されたバージョン及びアイテムに対して図１１に示されたテキスト形式のフォーマットを生成する目的から、前記オブジェクト・ノードによって確立されたデータ構造をトラバース（traverse）すべく使用されることが可能である。

【００９９】図１２のグラフィカル形式の表示ダイアログ・ウィンドウ150は、上述のXML型のルーチンからも生成される。当業者には容易に理解されるように、前記テキスト形式の情報を生成するために利用される同一のXML文書は、グラフィカル形式の表現をサポートすることが可能であり、或るオブジェクトがどの様に表示されるかについての知識とともに、前記ツリー構造における必要なオブジェクトを与えられる。限られた数の利用可能なオブジェクトの種類とともに、このような知識は、上記VCATシステム98によって提供される。例えば、或るアイテムが機能ブロックであることを前記XML文書が示すときに、上記VCATシステム98は、包括的に描写してある機能ブロックのルーチンを開始することが可能であり、そのようにする際に、その他のパラメータに従って機能ブロックを描くために、前記XML文書において提供されるデータを再び参照することが可能である。

【０１００】全てのアイテムがグラフィカル形式で表示されることがあり得るというわけでないと理解されるべきである。しかし、グラフィカル形式で表示され得るアイテムは、機能ブロック又は一連のフローチャート・アルゴリズムによって一般に規定されるものを含むが、これに限定するものではない。

【０１０１】前記監査証跡ダイアログ・ウィンドウ140におけるコンフィギュレーション・バージョンのユーザの選択から生じるグラフィカル形式の表示ダイアログ・ウィンドウ150は、前記コンフィギュレーションの閲覧を容易にするためにユーザに様々なオプションを提供する。詳述すれば、前記表示ダイアログ・ウィンドウ150は、上述の方法に従ってテキスト形式でフレーム152において表示されるコンフィギュレーション情報をユーザに表示させるテキスト表示ボタン206を有している。該テキスト表示ボタン206は、また、フレーム152において表示される特定の下位アイテムに案内するようにすることも可能である。詳述すれば、ユーザは、下位アイテムを選択した後で、テキスト形式で下位アイテムのコンフィギュレーション情報を見るために前記テキスト表示ボタン206を選択することが可能である。これに代えて、前記下位アイテムは、選択されるべきアイテムとして、テキスト表示のタスクを提供する状況メニュー207を生成すべく、右クリック等を使用して選択されることが可能である。

【０１０２】前記表示ダイアログ・ウィンドウ150は、親表示ボタン208及び掘り下げボタン210を更に有している。該掘り下げボタン210は、グラフィカル形式で元々選択されたアイテムの下位アイテムに対して、コンフィギュレーション情報を（可能な範囲で）表示するためのオプションをユーザに提供する。前記フレーム152において一旦表示されたアイテム（例えば、複合機能ブロック212）がユーザによって選択された場合には、ユーザがポインタ等を用いて前記掘り下げボタン210を選択することを可能とする。それから、該ボタン210は、関連するグラフィカル形式のコンフィギュレーション情報を表示するために、前記下位アイテムに「掘り下げ」ていくために選択されることが可能である。前記掘り下げタスクは、状況メニュー207を使用して開始されることも可能である。

【０１０３】前記親表示ボタン208は、下位アイテムの表示を削除することによって、前記親アイテムのグラフィカル形式の表示に戻るためのオプションをユーザに提供する。これに代えて、親表示操作が上記VCATシステム98に、前記下位アイテムに関連付けられた前記ダイアログ・ウィンドウを閉じさせ、それによって、前記親アイテムのグラフィカル形式の表示に戻らせるように、前記

下位アイテムは、個別のダイアログ・ウィンドウにおいて表示されることが可能である。

【０１０４】その他の実施形態において、グラフィカル形式の表示が下位アイテムに利用可能でない場合（例えば、それが複合機能ブロック又はモジュールを構成しない場合）には、テキスト表示ダイアログ・ウィンドウが生成される。グラフィカル形式の表示ダイアログ・ウィンドウ内で下位アイテムに掘り下げていく行為が、最終的にテキスト表示ダイアログ・ウィンドウという結果となると理解される。

【０１０５】バージョン制御データベース102及びユーザ・インタフェース94の生成の間の中間物としての高速に（on-the-fly）生成されたXML文書を利用することは、コンフィギュレーション情報の提示への効率的な且つ柔軟な手引きを提供するものと理解される。しかし、より重大なのは、各コンフィギュレーション・バージョンに対してXML文書を生成することが、選択されたアイテムの２つのバージョンの高速な且つ効果的な比較も提供することにある。

【０１０６】或るアイテムのバージョンを比較することは、何がバージョン間の「違い」と共通に称されるかを判定するためにバージョン制御の一般的な状況に対して有用である。以上に説明したプロセス制御システム10に関しては、テキスト形式及びグラフィカル形式の両方で、或るアイテムの２つのバージョンの違いを表示するために有益である。好ましい実施形態において、上記VCATシステム98は、バージョン制御データベース102に格納されたデータの上述したXML形式の処理を使用して、２つのバージョン間の違いについてテキスト形式及びグラフィカル形式の表示の両方のための違いダイアログ・ウィンドウを生成する。XML文書のツリー構造が前記オブジェクト・モデルに従って容易に解析されるので、或るアイテムの２つのバージョンは、対応する２つのXML文書を解析し、解析されたデータをオブジェクト単位で比較することによって、簡単に比較されることが可能である。

【０１０７】さて、図１６参照して、テキスト形式の違いダイアログ・ウィンドウ220は、第１のコンフィギュレーション・バージョン及び第２のコンフィギュレーション・バージョンに対するコンフィギュレーション情報を表示するための第１のフレーム222及び第２のフレーム224をそれぞれ有する。各フレーム222，224において表示されたテキスト形式の情報の水平方向及び垂直方向の両方へのスクロールは、公知のウィンドウズ手法を使用して提供される。前記第１のフレーム222は、古いバージョン（例えば、バージョン番号「13」）のコンフィギュレーション情報を記述することが可能である一方、前記第２のフレームは、新しいバージョン（例えば、バージョン番号「14」）のコンフィギュレーション情報を記述することが可能である。しかし、如何なる２つのバージョンも上記VCATシステム98によって比較され得る限り、前記バージョンが連続的に番号付けされる必要がないことを注記しておく。前記古いバージョンにあって、前記第１のフレーム222は、前記コンフィギュレーションのチェックアウト／チェックインの変更の結果として削除されたテキストの一又は複数の行を有することが可能である。このような削除された行に関連付けられたテキストは、残りのテキストに対して、異なる色（例えば、青），フォント，又はスタイルで示されることが好ましい。青色のテキストが、例えば、削除された行を表示するものであることを示すために、ボタン226が「削除された行」という青色のテキストを有することが可能である。同様に、前記フレーム224は、前記アイテムのコンフィギュレーションへの変更の結果として挿入された一又は複数の行のテキストを有することが可能である。テキストのこのような行は、例えば、「挿入されたテキスト」という緑色のテキストを有するボタン228とともに、緑色で記述されることが可能である。最後に、テキストの一又は複数の行の内容が、バージョン間で変更されていることがあり得る（但し、全体的に削除されるというわけではない）。このような行は、「変更された行」という赤色のテキストを有するボタン230とともに、前記フレーム222，224の両方において赤色で記述されることが可能である。任意の色又はスタイルの方式が、前述された差の種類を異ならせるために、また、変更されたテキストを両方のバージョンに共通のテキストから分離するために、利用され得ると理解される。

【０１０８】前記テキスト形式の違いダイアログ・ウィンドウ220は、テキスト表示ダイアログ・ウィンドウ146に関連して上述した同一の案内機能を提供する、検索ボタン232，下方へ検索ボタン234，上方へ検索ボタン236，下へボタン238，及び上へボタン240を有している。これらの案内ツールは、ユーザによって開始される如何なる垂直方向へのスクロール操作も両方のフレームをスクロールする結果となるように、フレーム222，224の両方に適用されていることが好ましい。その他の実施形態において、これらの案内ツールは、（ウィンドウ環境におけるその他の標準的なスクロール手法と同様に）例えば、ユーザによってどのフレームが選択されているかに基づいて、前記フレーム222，224のうちの１つに適用されることが可能である。

【０１０９】比較されているバージョンのためのテキスト形式の情報は、隣り合わせで記述される必要はない。例えば、テキスト形式の違いダイアログ・ウィンドウは、単一のフレームを代わりに有することが可能であり、共通のテキスト，変更されたテキスト；挿入された行，及び削除された行が、同様の色又はスタイルの方式を通じて表示及び区別される。前記２つのバージョンの違いは、赤線，下線，区切り記号等を使用してその状況によって示されることも可能である。

【０１１０】図１７は、グラフィカル形式の違いダイアログ・ウィンドウ250を示している。該グラフィカル形式の違いダイアログ・ウィンドウ250は、前記グラフィカル形式の表示ダイアログ・ウィンドウ150と同様の単一のフレーム・ウィンドウに基づいたものであるが、代わりに、バージョンを並列にすべく２つのフレームを隣り合わせにしたウィンドウを使用することが可能である。共通のオブジェクト，削除されたオブジェクト，追加されたオブジェクト，及び変更されたオブジェクトの間の違いを区別するために、色方式を再び採用することが可能である。色は、背景色，境界，マッチング，アウトライン等として前記オブジェクトに適用されることが可能である。ボタン252，254，及び256は、テキスト形式の違いダイアログ・ウィンドウ220に関連して上述したような同一の方法における色方式を基調として記述される。例えば、ステップ・アイテムS1及び移行アイテムT1は、該アイテムが変更されていることを示すために色付きの境界（例えば、赤）で示される。更なるステップ・アイテムS2及び更なる移行アイテムT2は、該アイテムが削除又は除去されていることを示すために異なる色付きの境界（例えば、青）で示される。

【０１１１】前記色方式が、また、前記アイテムを相互接続する線にも適用され得ることを注記しておく。

【０１１２】色付きのアウトライン又はフレームで示されたアイテムへの典型的な変更は、動作をステップに加えること、及び移行の表現を変更することを含んでいる。前記機能ブロックの実行命令への変更は、前記実行命令を表示することに供するオブジェクトの一部だけを、縁取り（アウトライニング），枠付け（フレーミング）等によって示されることが可能である。この部分は、例えば、オブジェクトの底部に位置することがあり得る。或るオブジェクトがバージョン間で名前を変更された場合に、前記オブジェクトの２つのバージョンが削除及び追加されたことを示すために、前記色方式を採用することも可能である。これに代えて、上記VCATシステム98は、表面的な変更が表示されないように、バージョン間の実質的な違いを表示するのみのオプションをユーザに提供することも可能である。

【０１１３】前記グラフィカル形式の違いダイアログ・ウィンドウ250は、選択されたアイテム及び任意の下位アイテムの両方に関して、前記ボタン206と同一の機能性を提供する表示テキスト・ボタン258を有している。

【０１１４】或る下位アイテムが変更されているように示される場合、上記VCATシステム98は、その下位アイテムに対してテキスト形式の違いダイアログ・ウィンドウ又は別のグラフィカル形式の違いダイアログ・ウィンドウの何れかを生成するために、前記アイテムを（その適切な選択を通じて）掘り下げる機能をユーザに提供する。１つの実施形態において、掘り下げることは、グラフィカル形式のダイアログ・ウィンドウが生成され得るそれらのアイテムに利用できるだけである。しかし、一般には、どの種類のダイアログ・ウィンドウが生成されたかは、前述した如く、アイテムの種類に依存する。下位アイテムへの掘り下げは、掘り下げボタン260の選択に伴って、開始されるようにすることも可能である。前のグラフィカル形式の違いダイアログ・ウィンドウ250への復帰は、ボタン208と同一の方法で動作する親表示ボタン262の選択によって達成されることも可能である。

【０１１５】除去されているアイテムが、追加されているアイテムと前記グラフィカル形式の違いダイアログ・ウィンドウの同一位置にあるとき、前記アイテムの１つを暗く表示することも可能である。加えて、コメントが変更されるときに、古いコメントは、グラフィカル形式の表示では見えないようにする。隠されたアイテム又はコメントをユーザに見せるために、上記VCATシステム98は、どのアイテム又はコメントが（例えば、最上位に）示されているかと、どのアイテムが隠されているかとを切り換えるメカニズム（ポインタ等を使用してメニュー・アイテム又は選択シーケンスを通じて）を提供することが可能である。

【０１１６】好ましい実施形態において、違い表示ダイアログ・ウィンドウを含む上述のテキスト形式及びグラフィカル形式の表示ダイアログ・ウィンドウは、テキスト形式及びグラフィカル形式の表示の間で切り換えることをユーザに許容するボタン又はタスク・アイテム・シーケンスを有している。上記VCATシステム98は、両方の形式で表示され得るアイテムに対して、ダイアログ・ウィンドウでこのような機能性を有するだけであることが好ましい。

【０１１７】上記VCATシステム98は、バージョン制御環境にセキュリティ・チェックを課すことがあり得ると理解される。例えば、或るレベル又は種類の認証を必要とする操作（例えば、ロールバック）にあっては、上記VCATシステム98は、前記操作を実行する前に、ユーザが前記操作を開始する許可を与えられているか否かを判定することが可能である。

【０１１８】新しいユーザが上記VCATシステム98に追加されるときに、上記VCATシステム98は、所望する程度の認証をユーザに提供することを選択され得る又は選択され得ない複数のVCATの操作に対応する複数のチェックボックスを有している新しいユーザ・ダイアログ・ウィンドウ（図示せず）を生成することが可能である。

【０１１９】上記VCATシステム98が、該VCATシステム98を有効又は無効にするオプションで適切な認証を有することをユーザに提供することが可能であると更に理解される。一旦無効とされた場合には、上記VCATシステム98は、前記チェックアウト／チェックイン及びその他の手順を課すことなく前記コンフィギュレーション・アプリケーション96を動作させる。再有効化されたとき、上記

VCATシステム98は、現在のコンフィギュレーション・データをバージョン制御データベース102に格納される最新のコンフィギュレーション・データと比較するために、コンフィギュレーション・データベース100にアクセスする整合ルーチンを実行することが好ましい。上記VCATシステム98が無効とされている期間に、コンフィギュレーションの変更に遭遇した各アイテムに対しては、新しいコンフィギュレーション・バージョンが、追加され、コンフィギュレーション・データベース100内の現在のバージョンを表わすデータが、それに関連付けられて格納される。新しいアイテムは、必要な範囲にバージョン制御データベース102において生成されることも可能でる。さらに、上記VCATシステム98が再有効化されていることを意味するために、ラベルを、バージョン制御データベース102内の全てのアイテムに割り当てることが可能である。

【０１２０】また、前記整合ルーチンは、何時でも実行されることが可能である。或る事例において、上記VCATシステム98は、バージョン制御データベース102を変更するために必要な範囲に、一又は複数のチェックアウトを元に戻す操作及び／又はチェックアウト操作を開始され得ると理解されるべきである。

【０１２１】上記VCATシステム98は、前記Explorerシステムによって管理されるコンフィギュレーション・データベース100をバックアップ及び回復するために利用されるルーチンと同様のデータベース・バックアップ／回復ルーチンを有することが可能である。その他の共通のデータベース・ユーティリティ、例えば、バージョン制御データベース102のデータ・ストレージ構造を修復することを促す「Clean Database」ルーチンは、上記VCATシステム98の一部として含まれることも可能である。

【０１２２】前記ユーザ・インタフェース94は、上記VCATシステム98及びコンフィギュレーション・アプリケーション96のユーザ及びオペレータにバージョン制御情報が提供される、唯一の出力デバイスでないことが好ましい。詳述すれば、一又は複数のルーチンは、プリンタ又はその他の提示デバイスに伝えるためのレポートの生成をサポートすべく、上記VCATシステム98によって実行されることが可能である。例えば、上記VCATシステム98は、どのアイテムがチェックアウトされるか、特定のアイテムの監査証跡、特定のユーザによってチェックアウトされたアイテムか、任意の削除された（但し、除去はされていない）アイテムか、そして、バージョン制御データベース102に格納されたデータに対する日付又はその他のパラメータでのチェックアウトのリストに関連する現在のバージョン制御情報のレポートを生成するようになされたタスク・アイテムを選択する機能をユーザに提供することが可能である。このような情報は、ローカル又はリモートに位置するか否かに拘わらず如何なる提示デバイスを介して提供されることが可能であり、或るフォーマットで又は任意のプロトコルに従って伝えられることが可能である。

【０１２３】このようなレポートの生成を容易にするために、上記VCATシステム98は、照会システムを有することが好ましい。該照会システムは、特定のユーザによる変更又は動作、指定された時間に発生した変更又はバージョン制御イベント、指定されたバージョン又はラベル内で発生した変更又はバージョン制御イベント、そして、特定のアイテム又はアイテムの範囲に関連する変更又はバージョン制御イベントのような、任意の数の対象を指向する検索基準を指定することを一般にユーザに許容する。ユーザが一又は複数の照会基準を入力した後で、上記VCATシステム98は、バージョン制御データベース102にアクセスし、そのコンテンツを解析する検索ルーチンを開始する。好ましい実施形態において、上記VCATシステム98は、その他のバージョン制御操作が開始され、それと並列実行され得るように、バックグラウンドで前記検索ルーチンを実行する。

【０１２４】前記照会システムは、ユーザ・インタフェース94を介して、表示用の一又は複数のダイアログ・ウィンドウを生成することが可能である。２つのこのようなダイアログ・ウィンドウの例は、図１８及び図１９に示されている。履歴レポート・オプション・ダイアログ・ウィンドウ280は、ドロップダウン・メニュー又は状況メニューを介してこれに指示されたタスク・アイテムの選択に続くアイテムの選択の後で、生成されることが可能である。前記履歴レポート・オプション・ダイアログ・ウィンドウ280は、生成されるべき履歴レポートがラベル及び下位アイテムに関するバージョン制御情報を有するか否かを促すチェックボックスを提供する。ユーザ・フィールド282は、指定されたユーザによって開始された変更又はバージョン制御イベントに案内するための履歴レポートも提供する。最後に、「何日から」フィールド284及び「何日まで」フィールド286は、履歴レポートに対する期間を指定するために使用され得る。前記期間の開始日付及び終了日付のエントリを容易にするために、ポップアップ式のカレンダー又は典型的な日付が、公知のウィンドウズ手法に従って各フィールド284，286に対して生成されたウィンドウに提供され得る。

【０１２５】図１９を参照して、前記照会システムは、或る検索基準を識別するための状態検索部292及び検索範囲部294を有する一般的な検索ダイアログ・ウィンドウ290を生成することが可能である。前記状態検索部292は、チェックされた全てのアイテム又は或るユーザにチェックアウトされたアイテムを検索することを上記VCATシステム98に指示するためのチェックボックスを有している。前記ユーザは、ドロップダウン・ウィンドウを介してユーザ名を提示する機能を有したユーザ・フィールド296によって指定されることが可能である。前記検索範囲部294は、選択されたアイテム、選択されたアイテ

ム及び全ての下位アイテム、又は、バージョン制御データベース102内の全てのアイテムのみに関するバージョン制御情報を検索することを、ユーザに許容する幾つかのチェックボックスを有している。

【０１２６】図５～図１９の画面表示は、その他の画面と同様に、標準的なウィンドウ式のコマンドでウィンドウ式のフォーマットを使用して生成及び変更されることが可能であるが、その他の任意のフォーマットも同様に使用されることが可能である。これらの画面表示のフォーマットは、例えば、上記VCATシステム98が、Explorerシステム又は前述したその他の特定のアプリケーション以外のコンフィギュレーション・アプリケーションに関連して使用されるような場合に、大幅に変更されることがあり得る。

【０１２７】ここで説明されたコンフィギュレーション・アプリケーション96及びVCATシステム98は、一又は複数のソフトウェア・ルーチンで実行されことが好ましいが、ハードウェア，ファームウェア等で具現化されるルーチンで実行されることが可能であり、プロセス制御システム10に関連付けられたその他の任意のプロセッサによって実行されることが可能である。従って、前述した操作及び手順の各々は、要求に応じて、標準的な多目的CPU又は特別に設計されたハードウェア又はファームウェアによって実行されることが可能である。ソフトウェアで実行されるとき、前記ソフトウェア・ルーチンは、磁気ディスク，レーザー・ディスク，又はその他の記録媒体、また、コンピュータ又はプロセッサ等のRAM又はROMの如き、コンピュータで読み取り可能な任意のメモリに格納されることが可能である。同様に、このソフトウェアは、例えば、コンピュータで読み取り可能なディスクによって、又はその他の可搬型のコンピュータ用記憶メカニズムによって、又は通信チャネル（例えば、電話線，インターネット等）を介して（これは、可搬型の記録媒体を介してこのようなソフトウェアを提供することと同一又は置き換え可能と見なされる）、等を含む任意の公知又は所望の伝送方法でユーザ又はプロセス制御システムに伝達されることが可能である。

【０１２８】このように、本願発明は、その説明の目的から、特定の例に関して記述されているものであって、本願発明を限定するものではない。また、本願発明においては、その目的及び範囲から逸脱することなく、開示されている実施形態の変更、追加、又は削除がなされ得ることは、当業者にとっては明らかである。

【０１２９】

【発明の効果】本願発明に係るプロセス制御システム及び方法においては、以上に詳述したように、プロセスの前記コンフィギュレーションの変更を示すデータを、前記変更に責任があるユーザのアイデンティティー，前記変更の時間及び日付，並びに前記変更の背景となる理由等の、前記変更に関する情報とともに記録することができる。

【図面の簡単な説明】

【図１】　複数のワークステーションを有するプロセス制御システムと、該複数のワークステーション上で実行されたプロセス・コンフィギュレーション・システムを利用するプロセスに指示し、該プロセスを設定するためのコントローラとを示すブロック図である。

【図２】　図１のワークステーションのうちの１つを介してユーザ・インタフェースを確立すべく前記プロセス・コンフィギュレーション・システムによって生成された典型的な画面表示である。

【図３】　図１のワークステーションのうちの１つを介していま１つのユーザ・インタフェースを確立すべく前記プロセス・コンフィギュレーション・システムによって生成された別の典型的な画面表示である。

【図４】　本願発明の１つの実施形態に係るバージョン制御及び監査証跡システムと統合される前記プロセス・コンフィギュレーション・システムを示すブロック図である。

【図５】　本願発明の多数の実施形態に係る前記プロセスの複数のバージョンのコンフィギュレーションに関する入力用及び出力用のユーザ・インタフェースを確立すべく、図４のバージョン制御及び監査証跡システムによって生成された画面表示である。

【図６】　本願発明の多数の実施形態に係る前記プロセスの複数のバージョンのコンフィギュレーションに関する入力用及び出力用のユーザ・インタフェースを確立すべく、図４のバージョン制御及び監査証跡システムによって生成された画面表示である。

【図７】　本願発明の多数の実施形態に係る前記プロセスの複数のバージョンのコンフィギュレーションに関する入力用及び出力用のユーザ・インタフェースを確立すべく、図４のバージョン制御及び監査証跡システムによって生成された画面表示である。

【図８】　本願発明の多数の実施形態に係る前記プロセスの複数のバージョンのコンフィギュレーションに関する入力用及び出力用のユーザ・インタフェースを確立すべく、図４のバージョン制御及び監査証跡システムによって生成された画面表示である。

【図９】　本願発明の多数の実施形態に係る前記プロセスの複数のバージョンのコンフィギュレーションに関する入力用及び出力用のユーザ・インタフェースを確立すべく、図４のバージョン制御及び監査証跡システムによって生成された画面表示である。

【図１０】　本願発明の多数の実施形態に係る前記プロセスの複数のバージョンのコンフィギュレーションに関する入力用及び出力用のユーザ・インタフェースを確立すべく、図４のバージョン制御及び監査証跡システムによって生成された画面表示である。

【図１１】　本願発明の多数の実施形態に係る前記プロ

セスの複数のバージョンのコンフィギュレーションに関する入力用及び出力用のユーザ・インタフェースを確立すべく、図４のバージョン制御及び監査証跡システムによって生成された画面表示である。

【図１２】　本願発明の多数の実施形態に係る前記プロセスの複数のバージョンのコンフィギュレーションに関する入力用及び出力用のユーザ・インタフェースを確立すべく、図４のバージョン制御及び監査証跡システムによって生成された画面表示である。

【図１３】　本願発明の多数の実施形態に係る前記プロセスの複数のバージョンのコンフィギュレーションに関する入力用及び出力用のユーザ・インタフェースを確立すべく、図４のバージョン制御及び監査証跡システムによって生成された画面表示である。

【図１４】　本願発明の多数の実施形態に係る前記プロセスの複数のバージョンのコンフィギュレーションに関する入力用及び出力用のユーザ・インタフェースを確立すべく、図４のバージョン制御及び監査証跡システムによって生成された画面表示である。

【図１５】　本願発明の多数の実施形態に係る前記プロセスの複数のバージョンのコンフィギュレーションに関する入力用及び出力用のユーザ・インタフェースを確立すべく、図４のバージョン制御及び監査証跡システムによって生成された画面表示である。

【図１６】　本願発明の多数の実施形態に係る前記プロセスの複数のバージョンのコンフィギュレーションに関する入力用及び出力用のユーザ・インタフェースを確立すべく、図４のバージョン制御及び監査証跡システムによって生成された画面表示である。

【図１７】　本願発明の多数の実施形態に係る前記プロセスの複数のバージョンのコンフィギュレーションに関する入力用及び出力用のユーザ・インタフェースを確立すべく、図４のバージョン制御及び監査証跡システムによって生成された画面表示である。

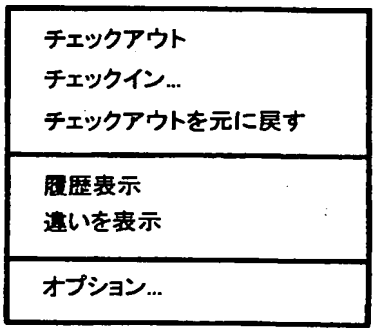
【図１８】　本願発明の多数の実施形態に係る前記プロセスの複数のバージョンのコンフィギュレーションに関する入力用及び出力用のユーザ・インタフェースを確立すべく、図４のバージョン制御及び監査証跡システムによって生成された画面表示である。

【図１９】　本願発明の多数の実施形態に係る前記プロセスの複数のバージョンのコンフィギュレーションに関する入力用及び出力用のユーザ・インタフェースを確立すべく、図４のバージョン制御及び監査証跡システムによって生成された画面表示である。

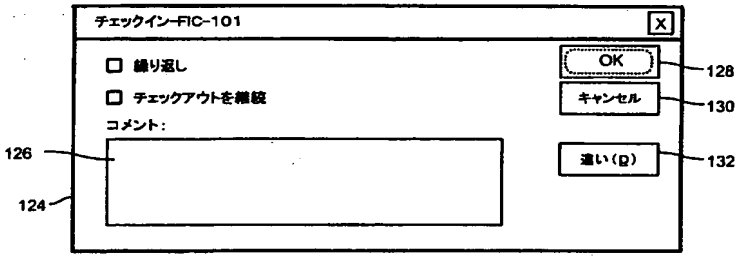
【符号の説明】

10　プロセス制御システム
12　プロセス・コントローラ
14　ホスト・ワークステーション
16　通信ネットワーク
18　プロセッサ
20　メモリ
24　プロセッサ
26　メモリ
30　Fieldbusデバイス・ネットワーク
32　HARTデバイス・ネットワーク
34　Profibusデバイス・ネットワーク
42　Fieldbusリンク
96　コンフィギュレーション・アプリケーション
98　VCATシステム（監査証跡システム）
100　コンフィギュレーション・データベース
102　バージョン制御データベース

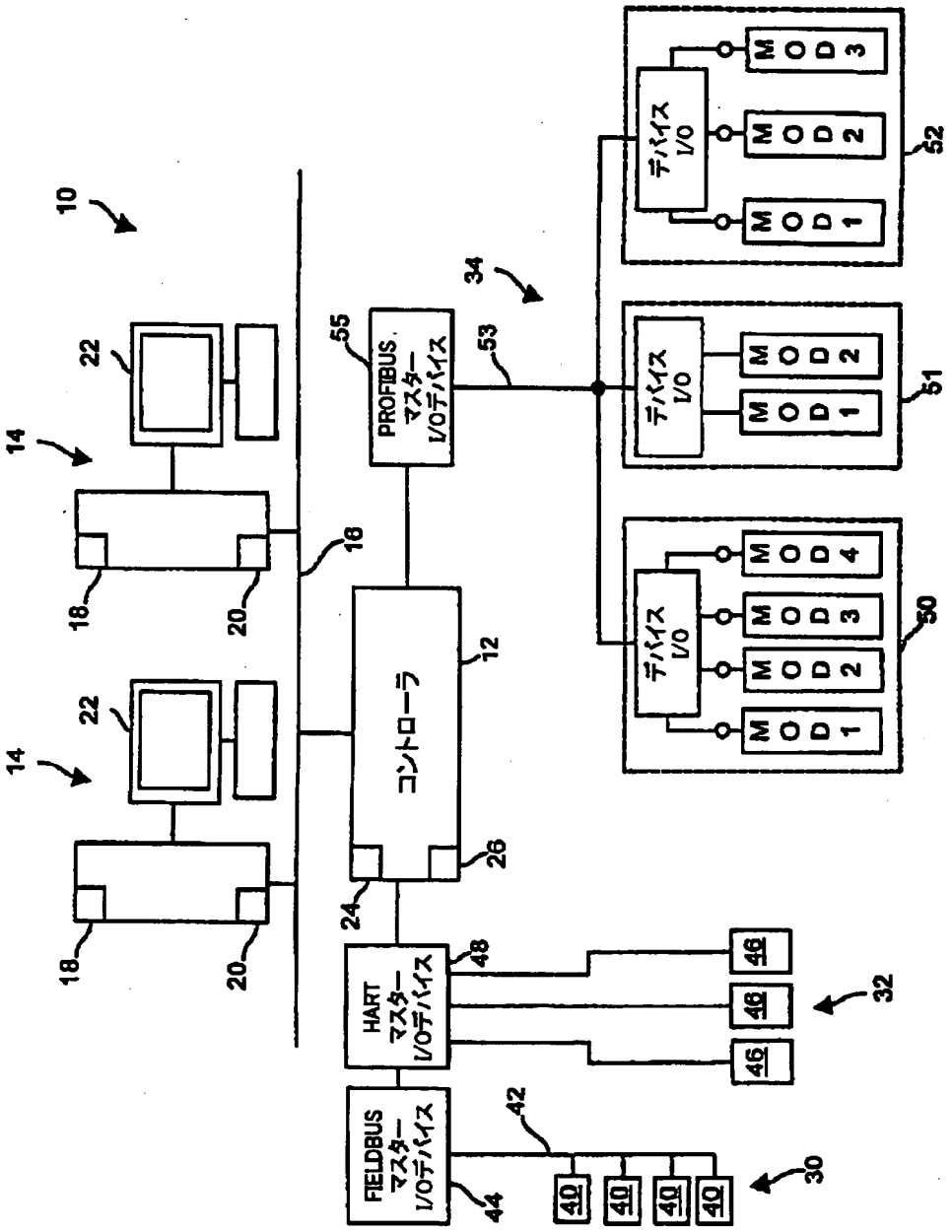
【図７】



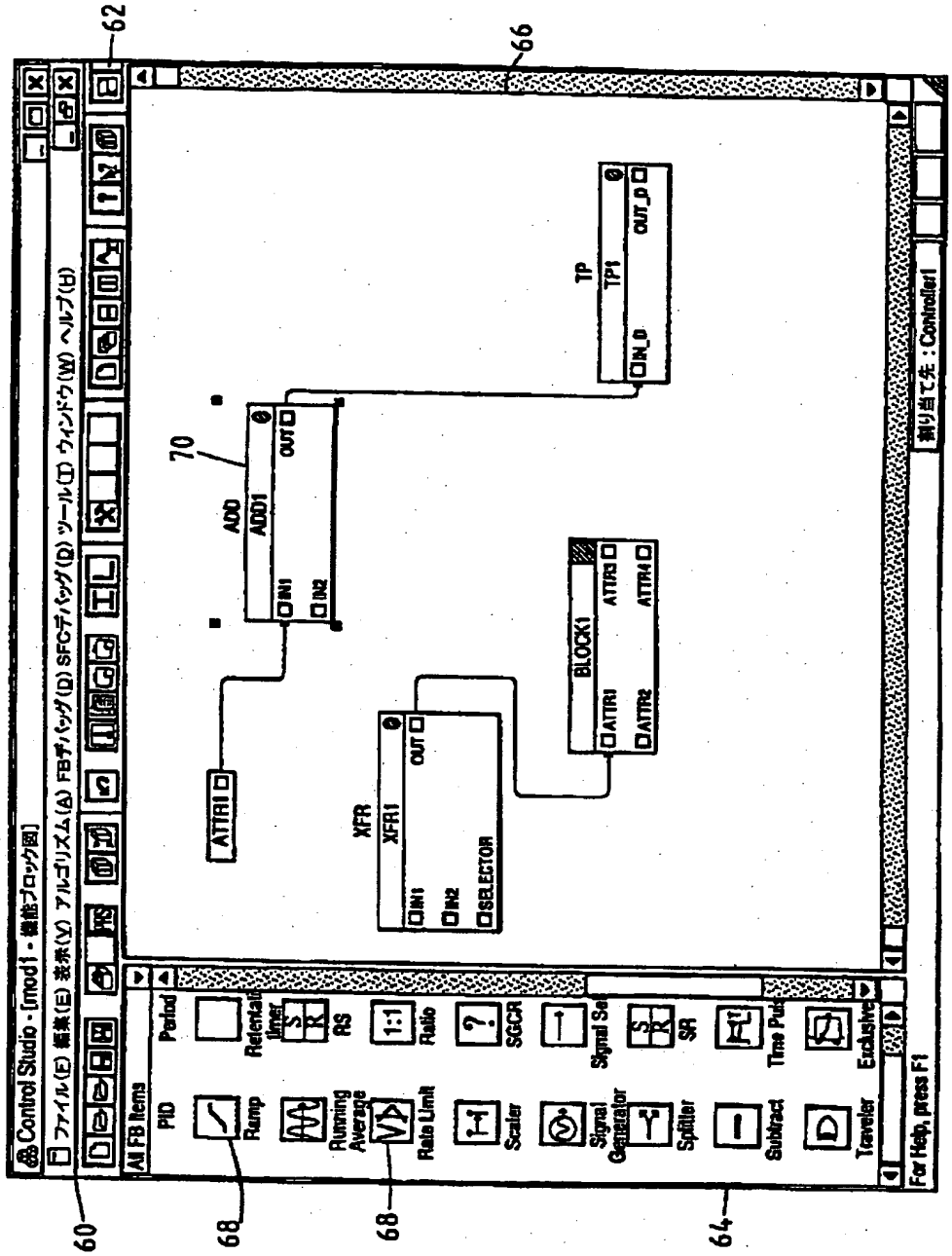
【図９】



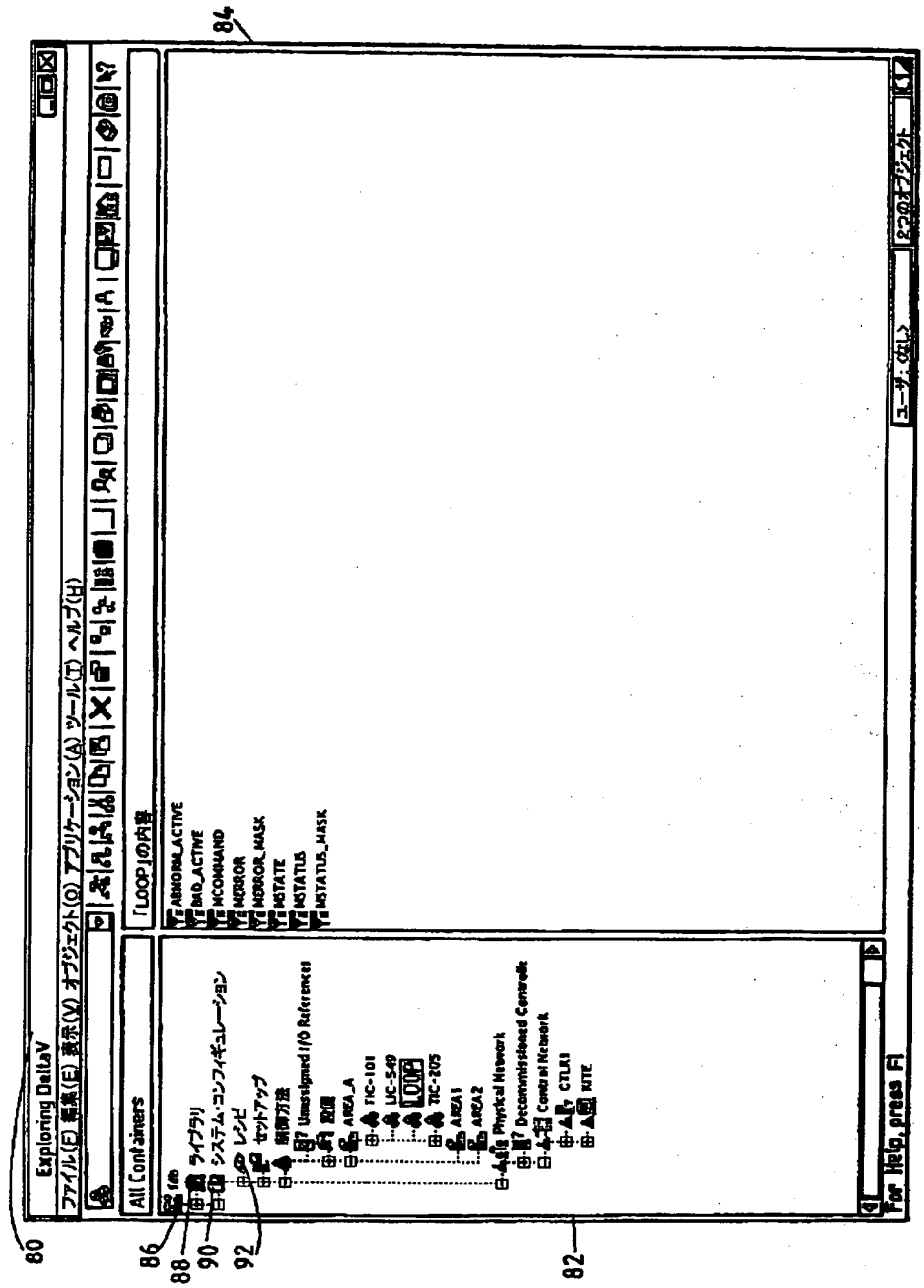
【図１】


10
14
22
18
20
16
12
コントローラ
24
26
34
PROFIBUS
マスター
I/Oデバイス
55
53
デバイス
I/O
MOD 1
MOD 2
MOD 3
MOD 4
52
51
50
HART
マスター
I/Oデバイス
48
46
32
FIELDBUS
マスター
I/Oデバイス
44
42
40
30

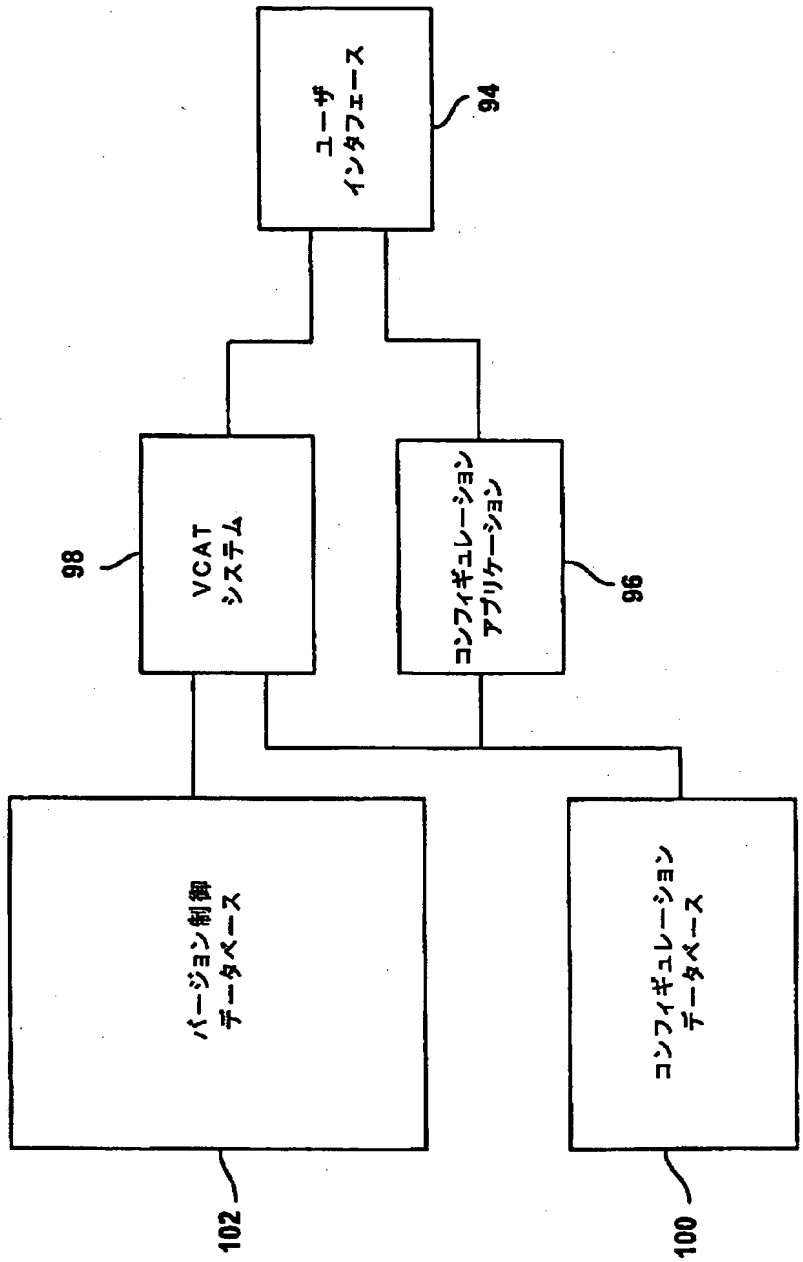
【図２】


Control Studio - [mod1 - 機能ブロック図]
ファイル(F) 編集(E) 表示(V) アルゴリズム(A) FBデバッグ(D) SFCデバッグ(D) ツール(T) ウィンドウ(W) ヘルプ(H)
All FB Items
PID
Ramp
Running Average
Rate Limit
Scaler
Signal Generator
Splitter
Subtract
Traveler
Exclusive
Time Put
SR
Signal Se
SGCR
Ratio
RS
Retentive timer
Period
For Help, press F1
割り当て先：Controller1
TP
TP1
OUT_D
IN_D
ADD
ADD1
OUT
IN1
IN2
ATTR1
XFR
XFR1
SELECTOR
BLOCK1
ATTR3
ATTR4
ATTR2
60
62
64
66
68
68
70

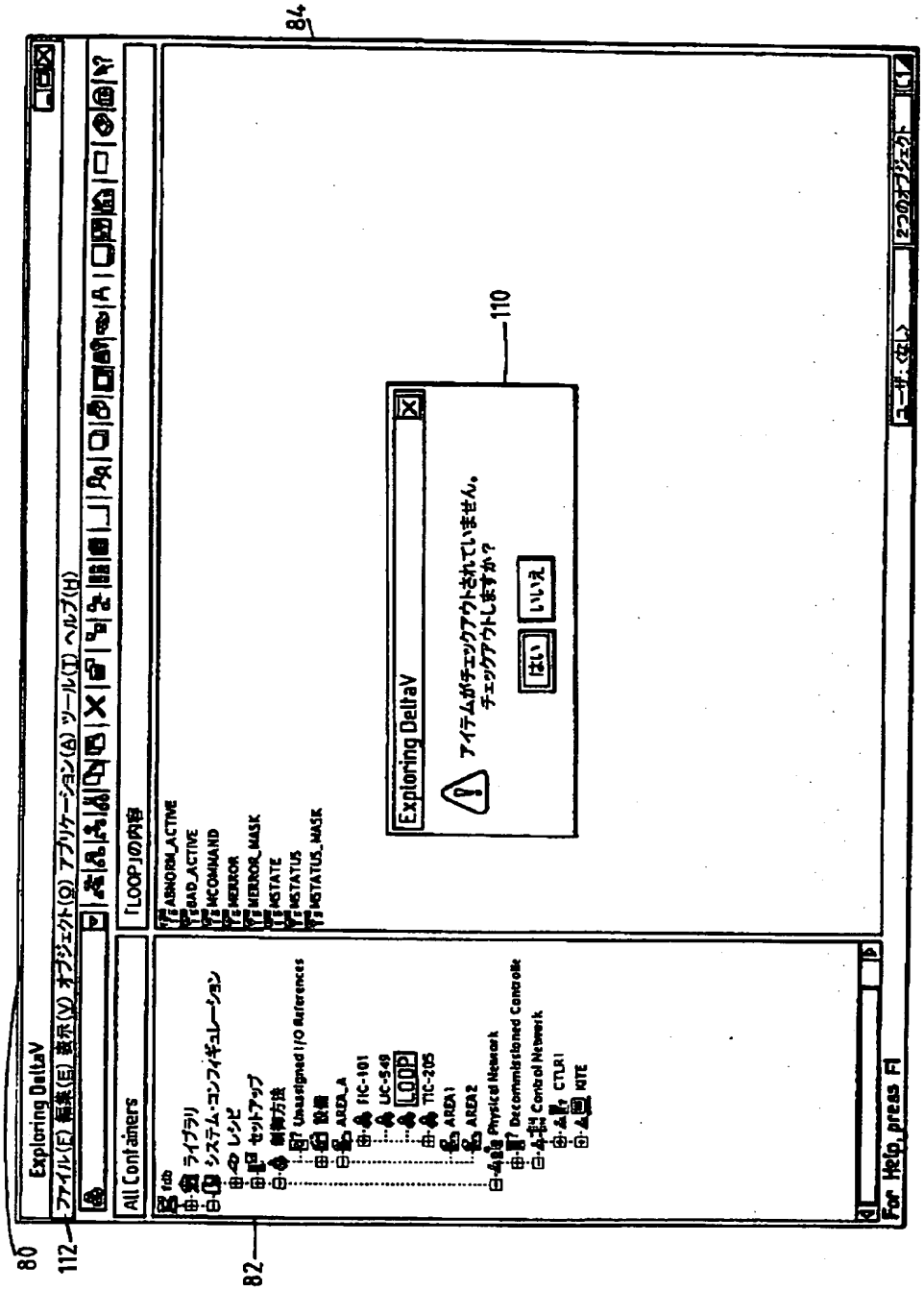
【図３】


Exploring DeltaV
ファイル(F) 編集(E) 表示(V) オブジェクト(O) アプリケーション(A) ツール(T) ヘルプ(H)
All Containers
「LOOP」の内容
ABNORM_ACTIVE
BAD_ACTIVE
MCOMMAND
MERROR
MERROR_MASK
MSTATE
MSTATUS
MSTATUS_MASK
ライブラリ
システム・コンフィギュレーション
レジン
セットアップ
Unassigned I/O References
AREA_A
TIC-101
LIC-540
LOOP
TIC-205
AREA1
AREA2
Physical Network
Decommissioned Controller
Control Network
CTLA1
NTE
For Help, press F1
80
82
84
86
88
90
92

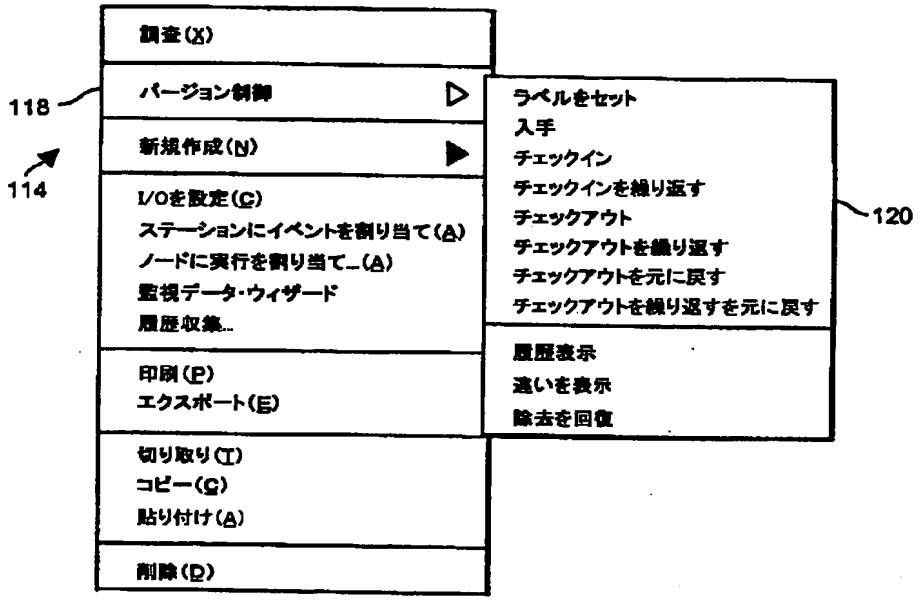
【図４】


ユーザ
インタフェース
94
VCAT
システム
98
コンフィギュレーション
アプリケーション
96
バージョン制御
データベース
102
コンフィギュレーション
データベース
100

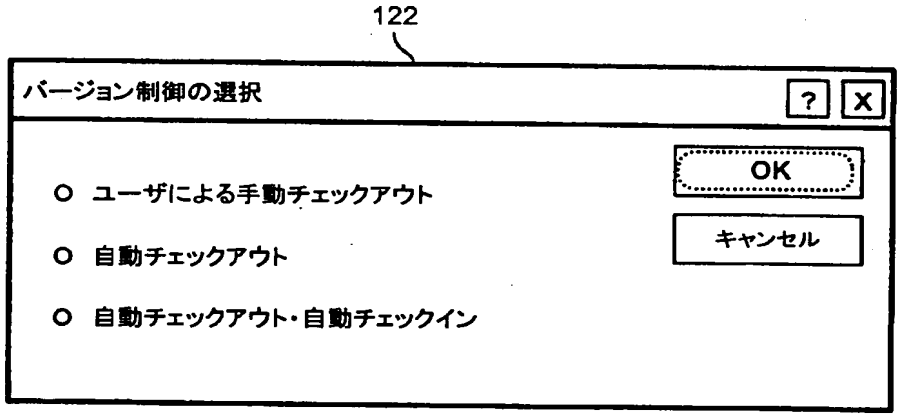
【図５】


80
112
82
84
110
Exploring DeltaV
ファイル(F) 編集(E) 表示(V) オブジェクト(O) アプリケーション(A) ツール(T) ヘルプ(H)
All Containers
「LOOP」の内容
ライブラリ
システム・コンフィギュレーション
レンピ
セットアップ
Unassigned I/O References
ARDA_A
FIC-101
LIC-549
LOOP
TIC-205
ARDA1
ARDA2
Physical Network
Decommissioned Controllers
Control Network
CTLR1
MTE
ABNORM_ACTIVE
BAD_ACTIVE
MCOMMAND
MERROR
MERROR_MASK
MSTATE
MSTATUS
MSTATUS_MASK
Exploring DeltaV
アイテムがチェックアウトされていません。
チェックアウトしますか？
はい
いいえ
For Help, press F1
2つのオブジェクト

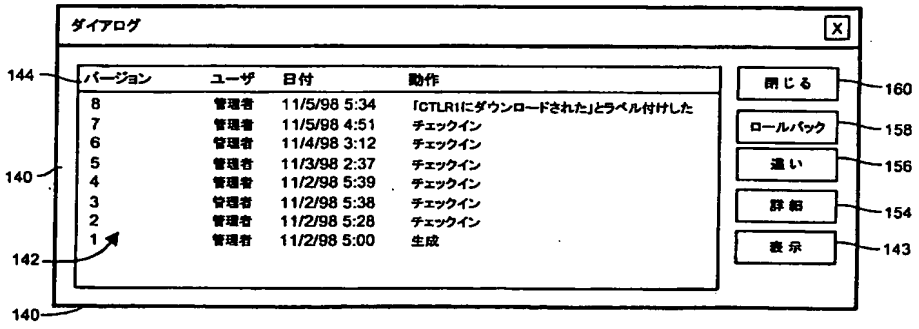
【図６】


調査(X)
バージョン制御
新規作成(N)
I/Oを設定(C)
ステーションにイベントを割り当て(A)
ノードに実行を割り当て...(A)
監視データ・ウィザード
履歴収集...
印刷(P)
エクスポート(E)
切り取り(T)
コピー(C)
貼り付け(A)
削除(D)
ラベルをセット
入手
チェックイン
チェックインを繰り返す
チェックアウト
チェックアウトを繰り返す
チェックアウトを元に戻す
チェックアウトを繰り返すを元に戻す
履歴表示
違いを表示
除去を回復
118
114
120

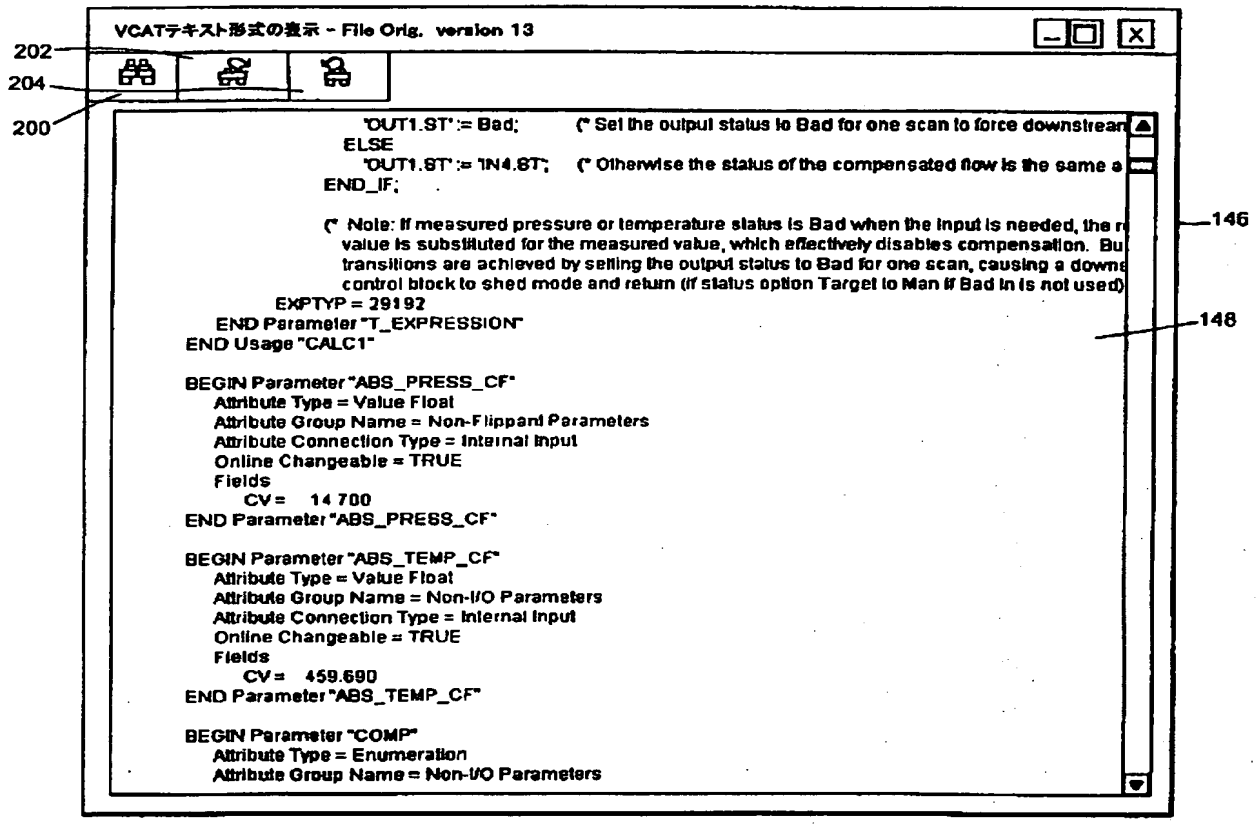
【図８】


122
バージョン制御の選択
?
X
OK
キャンセル
ユーザによる手動チェックアウト
自動チェックアウト
自動チェックアウト・自動チェックイン

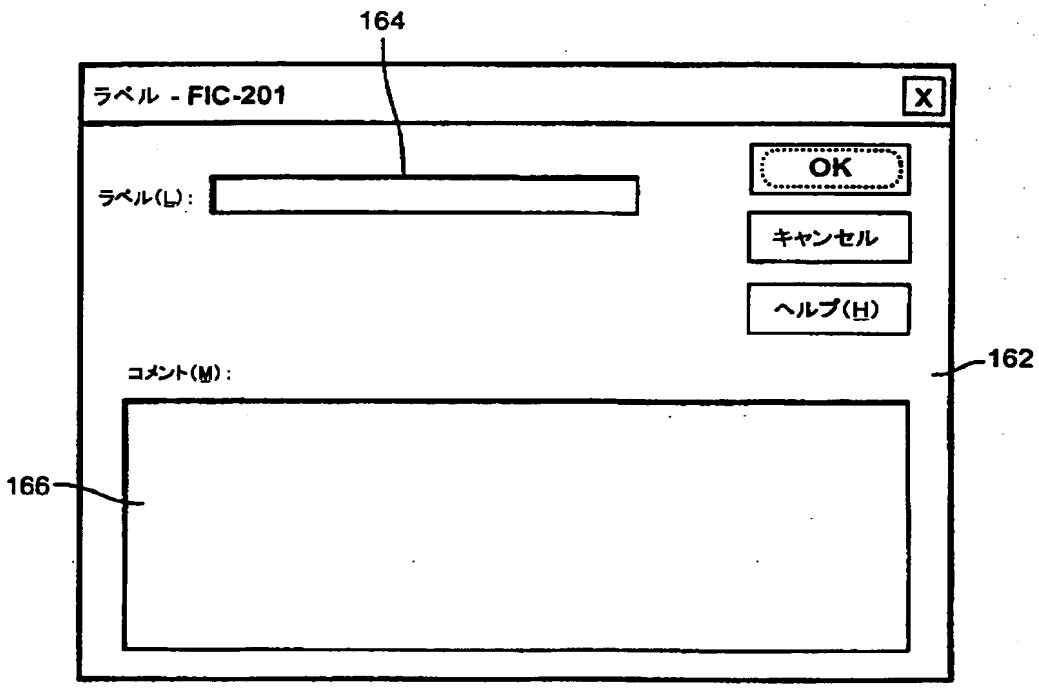
【図１０】


ダイアログ
X
144
バージョン
ユーザ
日付
動作
8 管理者 11/5/98 5:34 「CTLR1にダウンロードされた」とラベル付けした
7 管理者 11/5/98 4:51 チェックイン
6 管理者 11/4/98 3:12 チェックイン
5 管理者 11/3/98 2:37 チェックイン
4 管理者 11/2/98 5:39 チェックイン
3 管理者 11/2/98 5:38 チェックイン
2 管理者 11/2/98 5:28 チェックイン
1 管理者 11/2/98 5:00 生成
140
142
140
閉じる
ロールバック
違い
詳細
表示
160
158
156
154
143

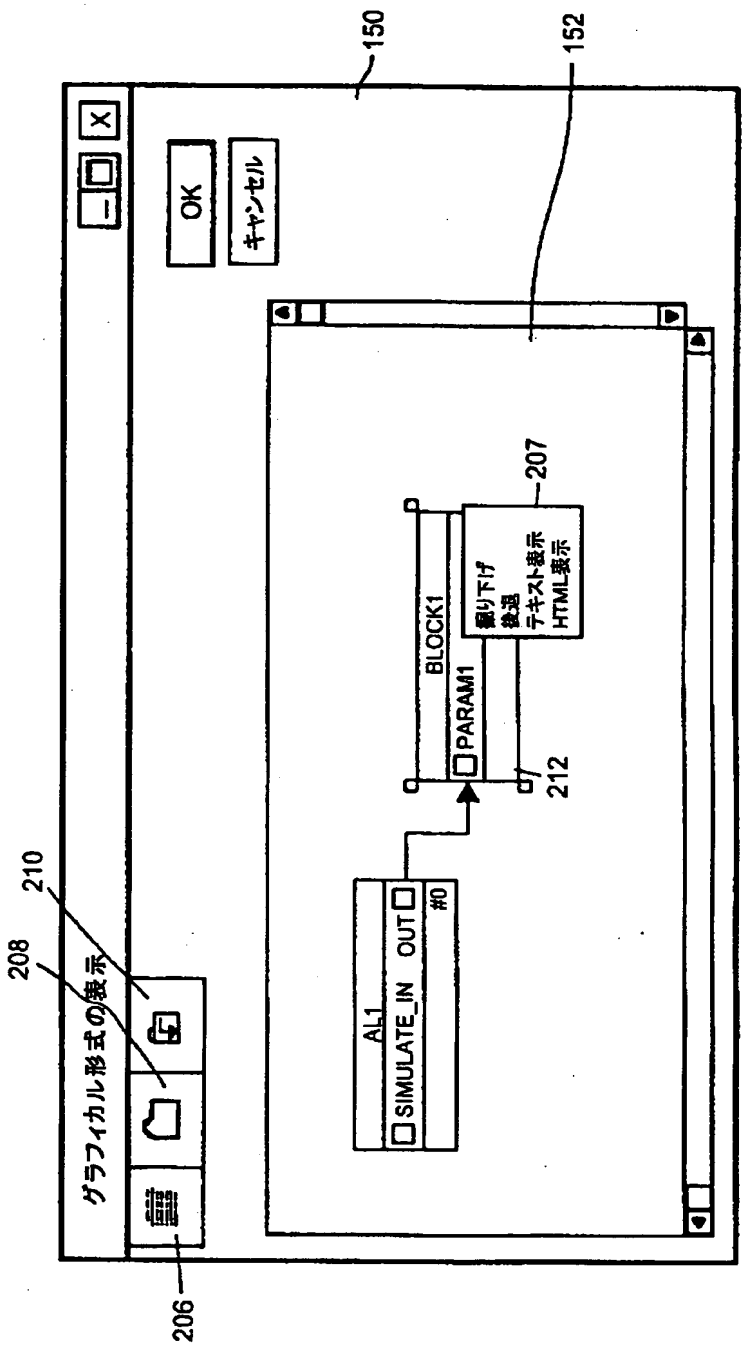
【図１１】


VCATテキスト形式の表示 - File Orig. version 13
202
204
200
'OUT1.ST' := Bad; (* Set the output status to Bad for one scan to force downstream
ELSE
'OUT1.ST' := 'IN4.ST'; (* Otherwise the status of the compensated flow is the same a
END_IF;
(* Note: If measured pressure or temperature status is Bad when the input is needed, the r
value is substituted for the measured value, which effectively disables compensation. Bu
transitions are achieved by setting the output status to Bad for one scan, causing a downs
control block to shed mode and return (if status option Target to Man if Bad In is not used)
EXPTYP = 29192
END Parameter "T_EXPRESSION"
END Usage "CALC1"
BEGIN Parameter "ABS_PRESS_CF"
Attribute Type = Value Float
Attribute Group Name = Non-Flippant Parameters
Attribute Connection Type = Internal Input
Online Changeable = TRUE
Fields
CV = 14.700
END Parameter "ABS_PRESS_CF"
BEGIN Parameter "ABS_TEMP_CF"
Attribute Type = Value Float
Attribute Group Name = Non-I/O Parameters
Attribute Connection Type = Internal Input
Online Changeable = TRUE
Fields
CV = 459.690
END Parameter "ABS_TEMP_CF"
BEGIN Parameter "COMP"
Attribute Type = Enumeration
Attribute Group Name = Non-I/O Parameters
146
148

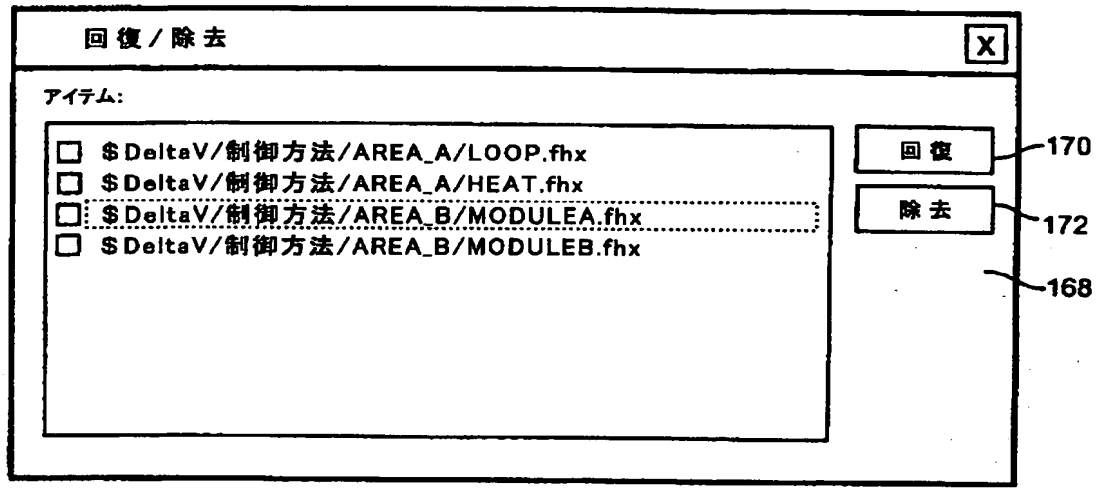
【図１３】


164
ラベル - FIC-201
ラベル(L):
OK
キャンセル
ヘルプ(H)
コメント(M):
162
166

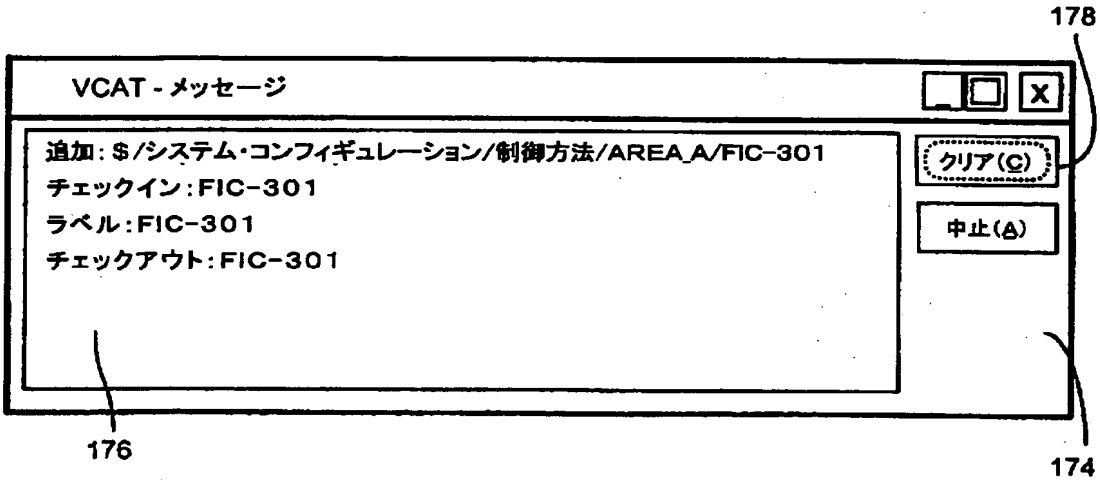
【図１２】


150
152
OK
キャンセル
207
BLOCK1
PARAM1
212
AL1
SIMULATE_IN
OUT
#0
210
208
206
グラフィカル形式の表示

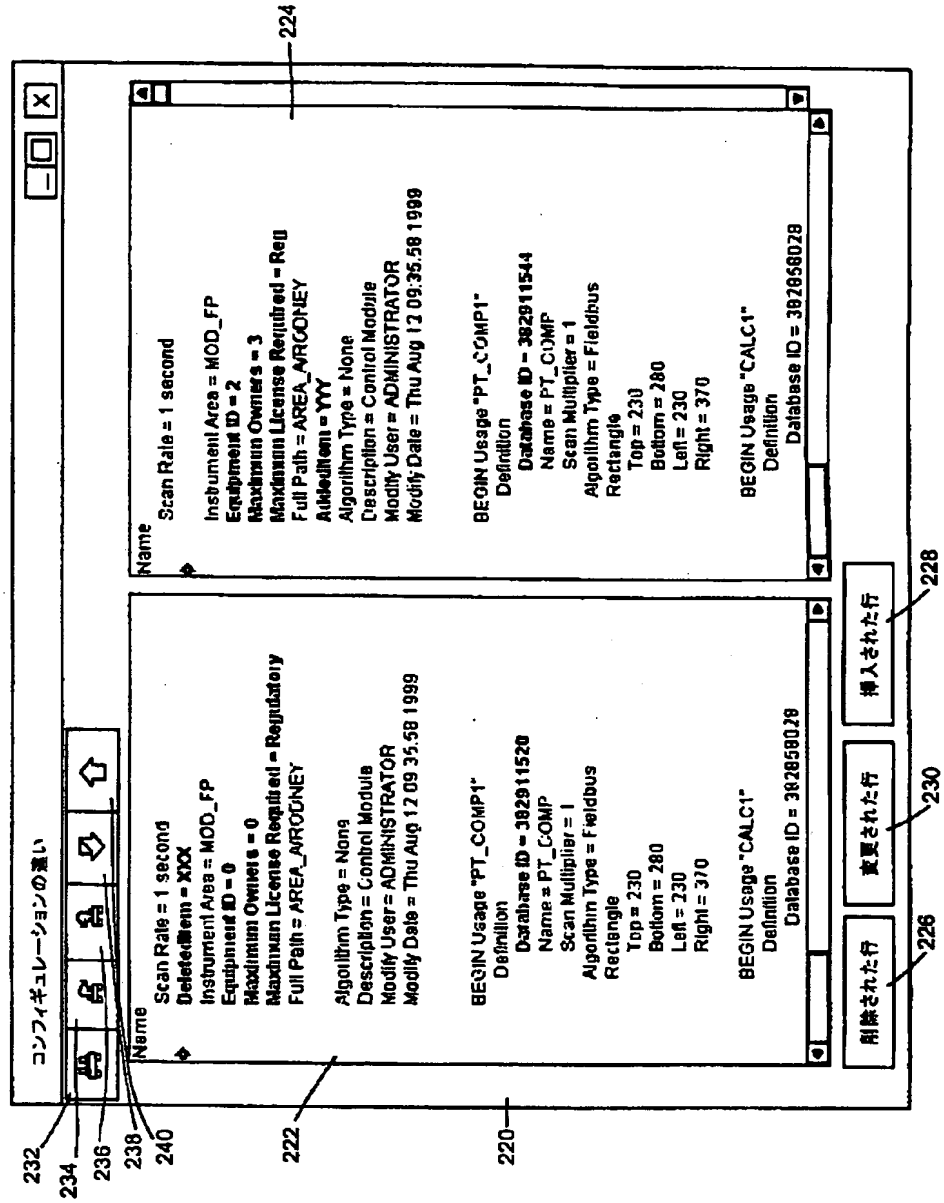
【図１４】


回復／除去
X
アイテム:
$DeltaV/制御方法/AREA_A/LOOP.fhx
$DeltaV/制御方法/AREA_A/HEAT.fhx
$DeltaV/制御方法/AREA_B/MODULEA.fhx
$DeltaV/制御方法/AREA_B/MODULEB.fhx
回復
除去
170
172
168

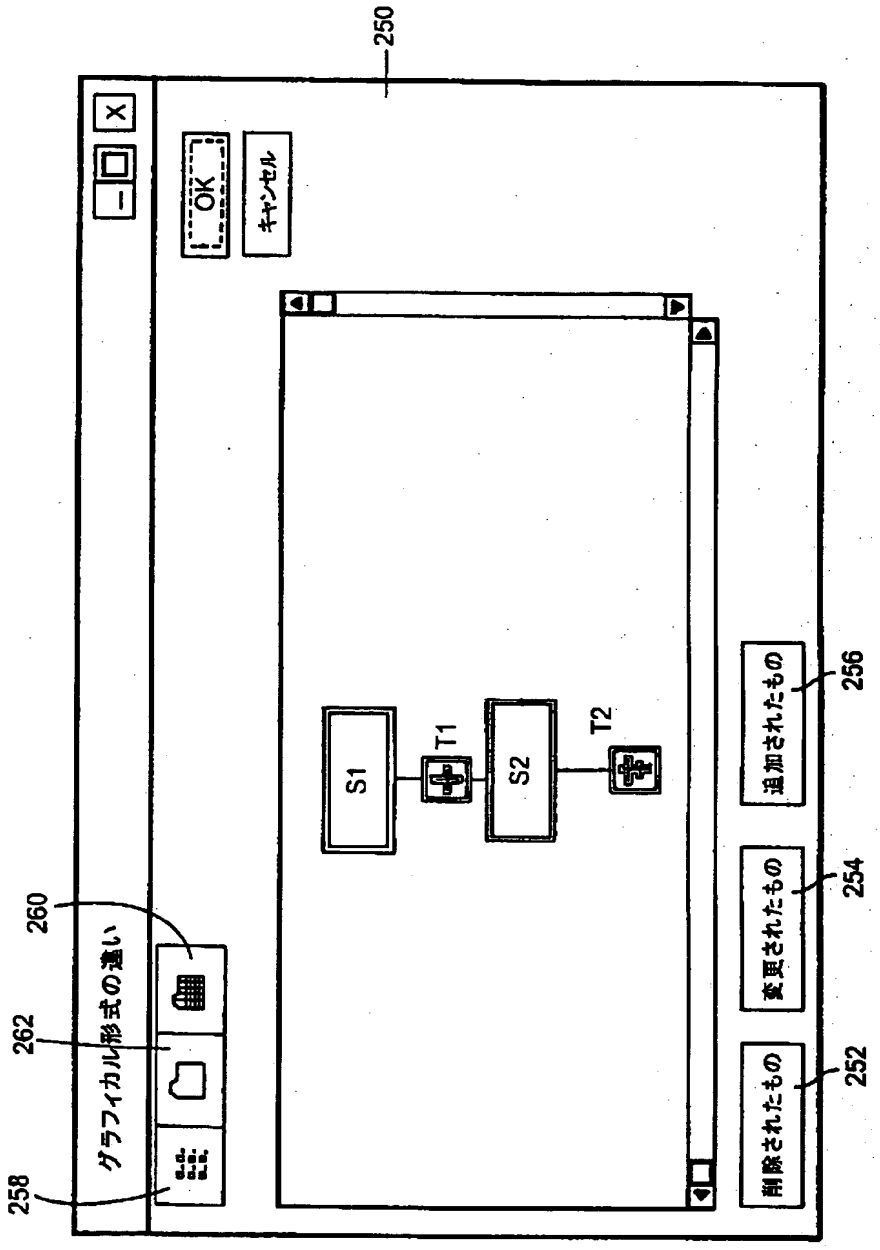
【図１５】


178
VCAT - メッセージ
X
追加: $/システム・コンフィギュレーション/制御方法/AREA_A/FIC-301
チェックイン: FIC-301
ラベル: FIC-301
チェックアウト: FIC-301
クリア(C)
中止(A)
176
174

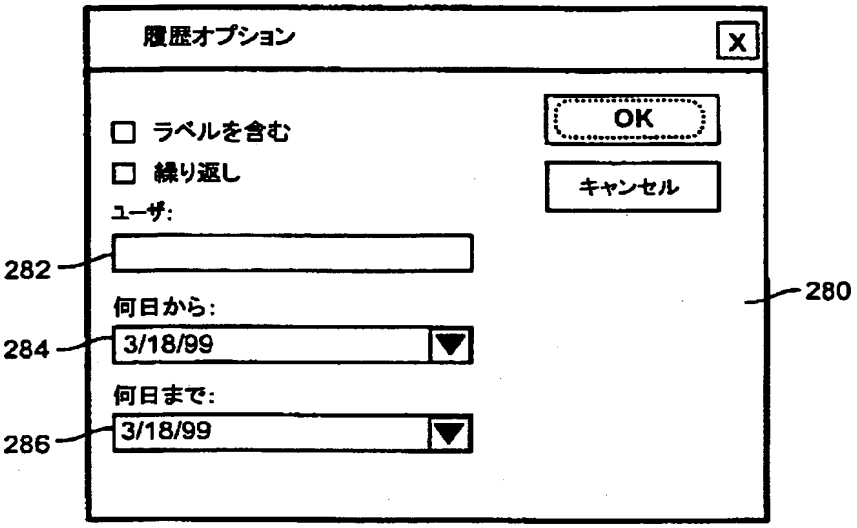
【図１６】


コンフィギュレーションの違い
Name
Scan Rate = 1 second
DeleteItem = XXX
Instrument Area = MOD_FP
Equipment ID = 0
Maximum Owners = 0
Maximum License Required = Regulatory
Full Path = AREA_A/ROONEY
Algorithm Type = None
Description = Control Module
Modify User = ADMINISTRATOR
Modify Date = Thu Aug 12 09:35:58 1999
BEGIN Usage "PT_COMP1"
Definition
Database ID = 382911520
Name = PT_COMP
Scan Multiplier = 1
Algorithm Type = Fieldbus
Rectangle
Top = 230
Bottom = 280
Left = 230
Right = 370
BEGIN Usage "CALC1"
Definition
Database ID = 382858028
Name
Scan Rate = 1 second
Instrument Area = MOD_FP
Equipment ID = 2
Maximum Owners = 3
Maximum License Required = Reg
Full Path = AREA_A/ROONEY
AllowItem = YYY
Algorithm Type = None
Description = Control Module
Modify User = ADMINISTRATOR
Modify Date = Thu Aug 12 09:35:58 1999
BEGIN Usage "PT_COMP1"
Definition
Database ID = 382911544
Name = PT_COMP
Scan Multiplier = 1
Algorithm Type = Fieldbus
Rectangle
Top = 230
Bottom = 280
Left = 230
Right = 370
BEGIN Usage "CALC1"
Definition
Database ID = 382858028
削除された行
変更された行
挿入された行
232
234
236
238
240
222
220
224
226
230
228

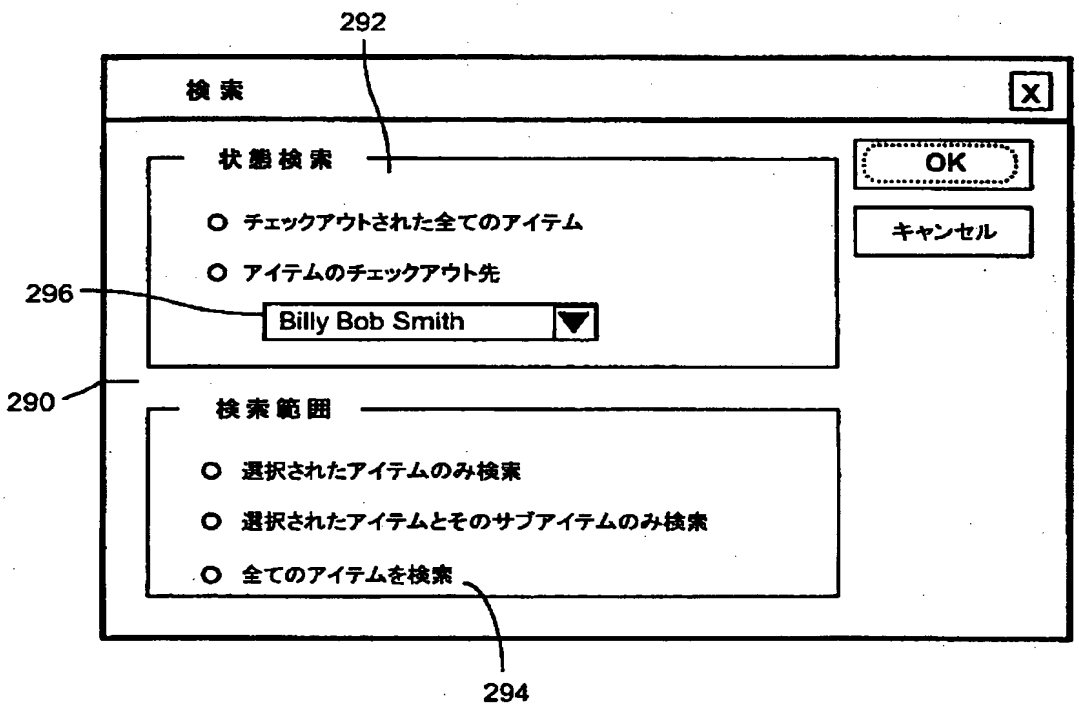
【図１７】


250
OK
キャンセル
S1
T1
S2
T2
追加されたもの
256
変更されたもの
254
削除されたもの
252
260
262
258
グラフィカル形式の違い

【図１８】


履歴オプション
X
ラベルを含む
繰り返し
ユーザ:
OK
キャンセル
282
何日から:
284
3/18/99
何日まで:
286
3/18/99
280

【図１９】


292
検索
X
状態検索
チェックアウトされた全てのアイテム
アイテムのチェックアウト先
296
Billy Bob Smith
OK
キャンセル
290
検索範囲
選択されたアイテムのみ検索
選択されたアイテムとそのサブアイテムのみ検索
全てのアイテムを検索
294

フロントページの続き

(72)発明者 ルーカス， マイケル ジェイ.
イギリス レイセスター エルイー９ ６
エヌエフ ブラウトン アストレイ ダー
ビー クローズ 10

(72)発明者 ダヴ， アンドリュー ピー.
アメリカ合衆国 78733 テキサス オー
スティン ティンバー リッジ パス
9803

【外国語明細書】

1 Title of Invention

VERSION CONTROL AND AUDIT TRAIL IN A PROCESS CONTROL SYSTEM

2 Claims

1. A system for controlling a process, the system comprising:

a computer-readable medium;

a processor in communication with the computer-readable medium;

a first database that stores first data representative of a first configuration of the process;

a second database that stores second data representative of a second configuration of the process;

a configuration routine stored in the computer-readable medium and configured to be executed by the processor to facilitate a modification of the first configuration of the process; and

a version control routine stored in the computer-readable medium and configured to be executed by the processor to store in the second database third data indicative of the modification of the first configuration of the process.

2. The system of claim 1, wherein the first database comprises a configuration database such that the first configuration of the process corresponds with a current configuration version.

3. The system of claim 1, wherein the second database comprises a version control database such that the second configuration of the process corresponds with a past configuration version.

4. The system of claim 1, wherein:

the first and second configurations of the process comprise first and second pluralities of process items, respectively; and

each process item of the first and second pluralities of process items has a respective item configuration such that --

the first configuration of the process comprises the item configuration of each process item of the first plurality of process items, and

the second configuration of the process comprises the item configuration of each process item of the second plurality of process items.

5. The system of claim 1, wherein the version control routine monitors the modification of the first configuration of the process to gather the third data.

6. The system of claim 5, wherein the version control routine monitors the modification by imposing a check-out/check-in procedure on the configuration routine.

7. The system of claim 6, wherein the check-out/check-in procedure is automatic.

8. A process control system having a process item, comprising:

a computer having a processor;

a process configuration application adapted to be implemented by the processor to establish a version of the item; and

a version control system in communication with the process configuration application to record and control a modification to the version of the process item.

9. The process control system of claim 8, wherein the process configuration application and the version control system are integrated such that the recordation and control of the modification to the version of the process item occurs in a manner transparent to a user.

10. The process control system of claim 8, wherein the modification is recorded and controlled in an automated manner.

11. The process control system of claim 8, wherein the version control system stores data indicative of a prior version of the process item.

12. The process control system of claim 11, wherein the version control system includes a comparison tool that provides a user interface for displaying differences between the version of the process item and the prior version of the process item.

13. The process control system of claim 12, wherein the user interface displays the differences graphically.

14. The process control system of claim 11, wherein the version control system includes a rollback tool to replace the version of the process item with the prior version of the process item.

15. The process control system of claim 14, wherein the rollback tool automatically determines whether any subordinate items of the process item have been deleted.

16. The process control system of claim 8, further comprising a process controller to which data reflective of the process item is downloaded wherein the version control system stores information indicative of the version of the process item to reflect when the data reflective of the process item is downloaded to the process controller.

17. The process control system of claim 8, further comprising a process controller to which data reflective of the process item is downloaded for implementation wherein the data includes information indicative of the version of the process item such that a process operator with access to the process controller may be informed of the version of the process item currently being implemented.

18. A method of controlling a process having a process item, the method comprising the steps of:

establishing a configuration of the process having a first version of the process item;

controlling a modification of the configuration of the process to create a second version of the process item; and

recording information associated with the modification of the configuration of the process.

19. The method of claim 18, wherein the recording step is performed in a transparent manner.

20. The method of claim 18, wherein at least one of the controlling and recording steps are performed in an automated manner.

21. The method of claim 18, further comprising the step of providing a user interface for displaying differences between the first version of the process item and the second version of the process item.

22. The method of claim 21, wherein the user interface displays the differences graphically.

23. The method of claim 18, further comprising the step of replacing the second version of the process item with the first version of the process item.

24. The method of claim 23, wherein the replacing step comprises the step of determining whether any subordinate items of the process item have been deleted.

25. The method of claim 18, further comprising the step of storing information indicative of a downloaded version of the process item to reflect when data reflective of the process item is downloaded to a process controller.

26. The method of claim 18, further comprising the step of downloading to a process controller information indicative of a current version of the process item such that a process operator with access to the process controller may be informed of the current version of the process item.

3 Detailed Description of the Invention

[Technical Field of the Invention]

The present invention relates generally to process control systems and, more particularly, to a system that monitors and records modifications to the process to provide version control therefor.

[Prior Art and Problems to be solved by the Invention]

Process control systems, like those used in chemical, petroleum or other processes, typically include at least one centralized process controller communicatively coupled to at least one host or operator workstation and to one or more field devices via analog and/or digital buses or other communication lines or channels. The field devices, which may be, for example, valves, valve positioners, switches, and transmitters (*e.g.*, temperature, pressure and flow rate sensors), perform functions within the process such as opening or closing valves and measuring process parameters. The process controller receives signals indicative of process measurements made by the field devices and/or other information pertaining to the field devices via an input/output (I/O) device, uses this information to implement a control routine and then generates control signals which are sent over the buses or other communication channels via the input/output device to the field devices to control the operation of the process. Information from the field devices and the controller is typically made available to one or more applications executed by the operator workstation to enable an operator to perform any desired function with respect to the process, such as viewing the current state of the process, modifying the operation of the process, and configuring the process.

Several software tools have been developed to assist the operator in configuring and otherwise modifying the process. Such tools provide a graphical representation of the process that displays each function performed by the field devices and the controller as a respective item or object. The items may be organized and set forth in a hierarchal environment such that, for instance, all of the functions performed by a particular field device are grouped or listed together. The items may

also be displayed within a control template such that items are shown in their functional relationships in the process. For example, a control template may constitute a sequential flow chart having a series of interconnected blocks representative of multiple items having input and output relationships defined by interconnecting lines.

Each item (as well as any input or output relationship between items) is defined by data stored in a configuration database associated with the software configuration tools. The database as well as the software tools are made available via a network that typically supports multiple workstations for numerous process operators and other users, each of whom has access to the data for administrative, process configuration, and other purposes. Modification of the data in the configuration database may, however, lead to version control problems when, for example, one operator is unaware of the work of another operator, or when too many modifications have occurred to determine how to return to a previous configuration or version of the process. For instance, a prior process control system utilizing the DeltaV™ software available from Fisher-Rosemont Systems, Inc., merely stores data indicative of the date on which the modification was made and the user responsible for the modification. Such data does not typically enable a user to reconstruct the configuration of prior versions of the process.

These version control concerns have been addressed in the software development context by configuration management tools, such as ClearCase from Rational and Microsoft Visual SourceSafe®. More particularly, these products track, control, and manage the development of a software routine in order to assist in subsequent debugging and development efforts. To this end, data indicative of both the current and past versions of the code is stored. Such configuration management tools, however, are usually limited to textual presentations of the data and, thus, are not well-suited for storing and displaying information that is typically set forth in a graphical manner.

A graphical interface has been developed by National Instruments to facilitate programming in connection with instrumentation systems. According to information

gathered from the National Instruments Internet website (www.ni.com), a product marketed under the name "LabVIEW" utilizes a graphical programming language, the G language, to support this graphical programming interface. The website further describes the LabVIEW product as including a development tool that allows the programmer to graphically compare the differences between two files of code set forth in the G language.

[Means for Resolving the Problems]

In accordance with one aspect of the present invention, a system useful for controlling a process includes a computer-readable medium, a processor in communication with the computer-readable medium, and first and second databases. The first database stores first data representative of a first configuration of the process, and the second database stores second data representative of a second configuration of the process. The inventive system further includes a configuration routine and a version control routine, both of which are stored in the computer-readable medium and configured to be executed by the processor. The configuration routine facilitates a modification of the first configuration of the process and the version control routine stores in the second database third data indicative of the modification of the first configuration of the process.

According to a preferred embodiment, the first database includes a configuration database such that the first configuration of the process corresponds with a current configuration version. The second database preferably includes a version control database such that the second configuration of the process corresponds with a past configuration version.

According to another preferred embodiment, the first and second configurations of the process include first and second pluralities of process items, respectively. Each process item of the first and second pluralities of process items has a respective item configuration such that the first configuration of the process includes the item configuration of each process item of the first plurality of process items, and

the second configuration of the process includes the item configuration of each process item of the second plurality of process items.

According to yet another preferred embodiment, the version control routine monitors the modification of the first configuration of the process to gather the third data. The version control routine may monitor the modification by imposing a check-out/check-in procedure on the configuration routine. Alternatively, the check-out/check-in procedure is automatic.

In accordance with another aspect of the present invention, a process control system having a process item includes a computer having a processor, a process configuration application adapted to be implemented by the processor to establish a version of the item, and a version control system in communication with the process configuration application to record and control modifications to the version of the process item.

In accordance with yet another aspect of the present invention, a method of controlling a process having a process item includes the steps of establishing a configuration of the process having a first version of the process item, controlling a modification of the configuration of the process to create a second version of the process item, and recording information associated with the modification of the configuration of the process.

[Embodiments of the Invention]

Processes typically require a great deal of ongoing design and development. As a result, it is often desirable to record data indicative of any modifications of the configuration of the process. For example, such configuration information is utilized in the pharmaceutical industries during monitoring by governmental agencies. As the complexity of the process (as well as the process control system) increases, it becomes more onerous to accurately determine the configuration of a prior version of the process.

The present invention provides a system and method for recording data indicative of modifications of the configuration of a process together with information relating to the modification, such as the identity of the user responsible for the modification, the time and date of the modification, and the reasoning behind the modification. The data is stored in accordance with the present invention as versions of the process configuration in a manner such that the inventive system and method enable a user to compare the differences between any two versions and return the process to a desired prior configuration version.

Referring now to FIG. 1, a process control system 10 includes a process controller 12 connected to one or more host workstations or computers 14 (which may be any type of personal computer, workstation, etc.) via a communication network 16 such as an Ethernet connection or the like. Each of the workstations 14 includes a processor 18, a memory 20 and a display screen 22. Similarly, the controller 12, which may be by way of example only, the DeltaV™ controller sold by Fisher-

Rosemont Systems, Inc., includes a processor 24 and a memory 26 for storing programs, control routines and data used by the processor 24 to implement control of a process. The controller 12 is coupled via the network 16 to numerous field devices within different device networks, including, for example, a Fieldbus device network 30, a HART device network 32, and a Profibus device network 34. Of course, the controller 12 could be connected to other types of field device networks such as 4-20 mA device networks and other local or remote I/O device networks in addition to or instead of the device networks illustrated in Fig. 1. The controller 12 implements or oversees one or more process control routines stored therein or otherwise associated therewith and communicates with devices within the device networks 30, 32, and 34 and with the host workstations 14 to control the process and to provide information pertaining to the process to one or more users, operators, process designers, etc.

In an exemplary case, the Fieldbus device network 30 includes Fieldbus devices 40 connected via a Fieldbus link 42 to a Fieldbus master I/O device 44 (commonly referred to as a link master device) which, in turn, is connected to the controller 12 via a local connection. Similarly, the HART device network 32 may include a number of HART devices 46 connected via communication lines to a HART master I/O device 48 which is connected to the controller 12 via a standard local bus or other communication line. The Profibus device network 34 is illustrated as including three Profibus slave devices 50, 51, and 52 connected via a Profibus link or bus 53 to a Profibus master I/O device 55. The Profibus master I/O device 55 may be in the form of a Profibus PCMCIA card attached to a standard I/O interface card.

Generally speaking, the process control system 10 of FIG. 1 may be used to implement batch processes in which, for example, one of the workstations 14 or the controller 12 executes a batch executive routine, which is a high level control routine that directs the operation of one or more of the field devices (as well as perhaps other equipment) to perform a series of different steps (commonly referred to as phases) needed to produce a product, such as a food product or drug. To implement different phases, the batch executive routine uses what is commonly referred to as a recipe which specifies the steps to be performed, the amounts and times associated with the

steps, the order of the steps, and the like. Steps for one recipe might include, for example, filling a reactor vessel (not shown) with appropriate materials, mixing the materials within the reactor vessel, heating the materials within the reactor vessel to a certain temperature for a certain amount of time, emptying the reactor vessel and then cleaning the reactor vessel to prepare for the next batch run. Each of these steps may define a respective phase of the batch run, and the batch executive routine within the controller 12 will execute a different control algorithm for each one of the phases. Of course, the specific materials, amounts of materials, heating temperatures and times, etc. may be different for different recipes and, consequently, these parameters may change from batch run to batch run depending on the product being manufactured or produced and the recipe being used.

The process control system 10 may also be capable of implementing process operations that are continuous in nature in addition to those initiated as part of a batch run. Thus, as used herein, the "process" should be understood to refer to, and include, any number of batch and/or continuous process operations executed or implemented by the process control system 10. During execution of the process, the process control system 10 is said to be in a run-time mode. The parameters and control algorithms to be utilized during run-time are defined while the process control system 10 resides in a configuration mode of operation. During the configuration mode (which may overlap in certain instances with periods of time in which process operations are being executed), one or more software applications executed on one or more of the workstations 14 are utilized to enable a user to specify the parameters, control algorithms, etc. for the process and, in so doing, generally design the configuration of the process.

With reference now to FIG. 2, the process control system 10 includes a process configuration system that is based on one or more process configuration applications, such as Control Studio™ and Recipe Studio™, which are both available from Fisher-Rosemont Systems, Inc. The process configuration applications, which may be implemented in software, are utilized to design either batch or continuous process operations (collectively referred to hereinafter as "the process"), or some portion

thereof. FIG. 2 is a screen display of a main control window of an exemplary process configuration application that enables a user to configure the process 10. The process configuration application of FIG. 2 is preferably executed on any one of the workstations 14, which may correspond with a centralized server (not shown) within the network 16. Alternatively, the application may execute in a distributed fashion such that more than one workstation 14 is responsible for implementation.

In general, the process configuration application provides a user interface that includes the main control window shown in FIG. 2. The main control window includes textual pull-down command menus 60, a pictographic command bar 62, a function library frame 64, and a flow diagram frame 66. Each of these portions of the user interface may be sized or positioned in accordance with typical windowing techniques. A plurality of icons 68 are displayed within the function library frame 64 each of which may be representative of a respective function block to be incorporated into a sequential flow chart created within the flow diagram frame 66. The sequential flow chart may, for example, define a control program to be executed during a phase of the process being designed. To build the control program, a user actuates one or more icons 68 by selecting the desired icon(s) in the library frame 64 with, for example, a pointing device (*e.g.*, a mouse), and dragging the actuated icons(s) to the flow diagram frame 66. The process configuration application then creates an object within the flow diagram frame 66, such as an addition block 70, that is representative of a function block. Data associated with the block 70 in the flow diagram frame 66 defines the function performed as well as whether the block 70 has any input ports, output ports, and the like, which are accordingly represented graphically as part of the block as shown. Function blocks and other objects added to the sequential flow chart (such as an input parameter) are coupled with lines drawn by the user (with a drawing tool selected, for example, from the command bar 62) in the flow diagram frame 66 to establish the data relationships between the objects. Information relating to the time sequence in which each function block is to be executed may also be specified by the user and shown in connection with the displayed object.

The data representative of each function block, phase, etc. is stored in a configuration database, which is accessed and developed by the process configuration application. Practice of the present invention is not limited to any particular type of database and, thus, the configuration database may take on any form or structure. For example, the data stored in the configuration database need not be stored in a local or localized manner such that the data stored in the configuration database is distributed across the network 16. The configuration database, however, preferably comprises an object-oriented database located in the memory 20 of one of the workstations 14 for storing data representative of the process configuration, which includes, for example, the relationships between the function blocks assigned within the sequential flow chart designed in the flow diagram frame 66.

Once the sequential flow chart has been designed using the above-described functionality, the resulting control program may be stored as a composite item of the process configuration. Composite items developed by the Control Studio™ software system may, in turn, be utilized to build modules. To that end, data representative of a module may also be stored in the configuration database. Both composite items and modules may be represented using a sequential flow chart.

Further examples of items that may be represented using a sequential flow chart include recipes, procedures, and unit procedures. These items may be designed by another portion of the process configuration system, the Recipe Studio™ application, to be representative of a batch run (or portion thereof) to be executed as part of the process. The Recipe Studio™ application generates a user interface similar to that shown in FIG. 2, and may provide a library frame having icons representative of modules as well as function blocks. Data representative of the recipes, procedures, and unit procedures may then also be stored in the configuration database.

Together, the data representative of these function blocks, modules, recipes, etc. may define the process as it is currently being implemented by the process control system 10. To this end, the configuration database is accessed and utilized when instructions are downloaded to the controller 12 and the field devices.

With reference now to FIG. 3, the data stored in the configuration database may be presented to a user via a configuration database administrative interface such as DeltaV® Explorer, which will hereinafter be referred to as "the Explorer system" with the understanding that any database administrative interface may be utilized. The Explorer system sets forth a configuration hierarchy in a windows-type environment having a suite of configuration tools for modifying the elements of the hierarchy. More particularly, a main window 80 developed by the Explorer system includes a hierarchy frame 82 in which the configuration hierarchy is displayed and a contents frame 84 in which additional information for each component of the hierarchy is displayed. The hierarchy frame 82 includes a plurality of icons that identify various levels of the hierarchy, such as a top level 86 representative of the entire process configuration, as well as two secondary levels representative of a library folder 88 and a system configuration folder 90. The library folder 88 may include the information utilized during process design in which function blocks and/or modules are applied within a phase or process as part of a control program. The system configuration folder 90 then includes the results of the design procedures in one or more folders therein, such as a recipes folder 92. Each folder beyond the secondary level may be expanded or contracted within the hierarchy frame in accordance with known windowing techniques. Alternatively, the components of a folder may be displayed in the contents frame 84 by selecting the name of the folder using a pointer or the like. For example, the contents of a control program LOOP associated with AREA_A utilizes a number of items or objects, the names of which are identified in the contents frame 84. Selecting one of these items may provide the user with information relating to properties of the item (*e.g.*, object type, application used to develop the item, data storage location, etc.), as well as a graphical display of a sequential flow chart in a process configuration application (*e.g.*, Control Studio) or a textual display if no sequential flow chart exists.

It should be noted that the tools provided by the process configuration applications (*e.g.*, Control Studio™ and Recipe Studio™) and the Explorer system described in connection with FIGS. 2 and 3, respectively, may be integrated into a

single application to any desired extent. Furthermore, the exact manner in which the configuration of the process is accomplished is not pertinent to the practice of the present invention. Thus, for purposes of simplicity in explanation only, as used hereinafter, the "configuration applications" should be understood to include the process configuration applications described in connection with FIG. 2 as well as the Explorer system described in connection with FIG. 3.

With reference to FIG. 4, a user interface 94 is generated on the display 22 of one of the workstations 14 (FIG. 1) to enable a user to implement one or more configuration applications 96. In accordance with the present invention, the user interface 94 is also generated in connection with a version control and audit trail system 98 (hereinafter "the VCAT system"), which, in general, cooperates with the configuration applications 96 to record and administer historical information regarding the configuration of the process. Both the configuration applications 96 and the VCAT system 98 access and otherwise communicate with a configuration database 100, which, as described hereinabove, stores data representative of the current configuration of the process. The VCAT system 98 is also in communication with a version control database 102, which is administered thereby in accordance with the present invention.

The version control database 102 includes configuration history data indicative of any number of prior versions of each item utilized in the process configuration. Taken together, the history data for all of the items may be used to reconstruct past configurations of the process. More particularly, for each item in the configuration database 100 (as well as those no longer in the configuration database 100), data representative of the configuration of that item is stored for a plurality of versions. For example, an item may have been modified on three occasions since it was created. The version control database 102 would therefore have data indicative of the configuration of the item at the point of creation, which may be referred to as "Version 1," as well as data indicative of the configuration of the item after each of the three modifications, which would correspond with "Version 2," "Version 3," and "Version 4."

The configuration history data is thus representative of all of the modifications made to the function blocks, modules, phases, recipes and any other aspects of the process configuration. The modifications may, but need not, be made using the configuration applications 96. In such a case, however, practice of the present invention is preferably accomplished by integrating the functionality of the VCAT system 98 into the user interface 94 generated by the configuration applications 96, as shown schematically in FIG. 4 and as explained in greater detail hereinbelow. To this end, the configuration applications 96 and the VCAT system 98 may, but need not, be combined into a single, integrated system. To the extent necessary for clarity, however, tasks executed in accordance with the present invention will be attributed to the configuration applications 96 and the VCAT system 98 separately.

The VCAT system 98 is preferably implemented using one or more of the workstations 14 in a manner that allows for the monitoring of any modifications to the process configuration. Such monitoring may be furthered by the integrated system described hereinabove. Alternatively, the VCAT system 98 may be executed on a workstation or other device that does not correspond with the workstation 14 utilized by the operator or process designer, but which is in communication with the process control system 10 for recording data indicative of the configuration modifications.

The data in one or both of the databases 100, 102 may be stored in a computer-readable medium physically located anywhere within the process control system 10, such as, for example, the memory 20 or a magnetic or optical storage medium associated with one of the workstations 14. Alternatively, one or both of the databases 100, 102 may be stored in a remote location such that the workstation 14 accesses the data stored therein over a network such as an intranet, the Internet, or any other communication medium. Furthermore, the data stored in each database 100, 102 need not be stored in the same computer-readable medium, such that any portion of either database 100, 102 may be stored in a memory device or medium which is distinct from devices or media storing other portions.

In FIG. 4, the VCAT system 98 is shown as distinct and separate from the version control database 102. In an alternative embodiment, the version control

database 102 forms a portion of the VCAT system 98. Similarly, the configuration database 100 and the version control database 102 may, but need not, constitute separate and distinct data structures. That is, the databases 100, 102 may be located in the same storage medium and, in fact, may compose portions of a common database dedicated to the process control system 10. Accordingly, a "database," as used herein, should be understood to not be limited to any particular data structure.

In one embodiment, the version control database 102 comprises a relational database. Alternatively, the version control database 102 may be generated using a reference application that provides version control tools, such as Microsoft Visual SourceSafe®, as the version control data repository. In yet another alternative embodiment, the version control database 102 may be file-based.

FIG. 5 is a screen display for the user interface 94 similar to that shown in FIG. 3 but incorporating functionality provided by the VCAT system 98 in an integrated fashion with the configuration applications 96. In such an integrated system, one or more tools have been added to the configuration applications 96 for implementation and control of the VCAT system 98 in the context of the administration of the process control system 10. For example, when the VCAT system 98 is enabled (as explained in greater detail hereinbelow), and a user attempts to view and/or modify an item administered by the configuration applications 96 by double-clicking or otherwise selecting it, a dialog 110 is generated within the user interface 94 that alerts the user of the need to "check-out" the item from the configuration database 100 before proceeding. Similarly, when a user adds a new item to the process configuration, such an integrated system automatically adds the new item to both the configuration database 100 as well as the version control database 102, and checks out the item to facilitate its creation within the configuration applications 96.

In short, the VCAT system 98 preferably imposes a check-out/check-in environment on the design of the process. In this environment, an item must be checked-out (either manually or automatically) before modifications may be made so that the VCAT system 98 may record information relating to the item and, in so

doing, generally prepare for an upcoming "check-in" operation. If, for instance, the item "MCOMMAND" shown in the contents frame 84 requires modification, the user would have selected the item in the contents frame 84, and selected a "YES" option within the dialog 110, after which the configuration database 100 storing the current version of the process is accessed to retrieve the data associated with that item. The retrieved data is representative of information that may be displayed in one or more ways, such as textually or graphically. Thus, functionality provided by the configuration applications 96 (*e.g.*, Control Studio™) may be necessary to view or modify the retrieved information. In any event, the appropriate application is utilized to view the information and make any desired modifications, at which time the user would typically execute a Save task.

It should be noted that the functionality provided by the configuration applications 96 related to merely viewing the configuration of an item need not be affected by the integration with the VCAT system 98. That is, the VCAT system 98 preferably allows the user to view a "read-only" copy of the configuration information without requiring or initiating the check-out/check-in procedure.

The initiation of a Save task to store such modified information associated with the item preferably precedes the check-in of the item. More particularly, execution of the Save task first causes the configuration applications 96 to store data representative of the modified information associated with the item in the configuration database 100. Next, the VCAT system 102 begins execution of the check-in task to store data representative of the modified information in the version control database 102. Execution of the check-in task may include the generation of another dialog (not shown) that allows the user to enter a comment regarding the check-in operation. The comment may, for instance, be directed to the reason behind the modification to the configuration. Data representative of the comment is then stored in the version control database 102 and associated with the data representative of the modified configuration.

The check-out/check-in environment is preferably limited to only those items in the configuration database 100 that are "versionable." In general, a versionable

item should be understood to include any item for which historical configuration information is maintained or stored in association with the item itself. In a preferred embodiment, versionable items include those items the configuration of which has been designed, such as any module, composite function block, etc. Although other items, such as a module parameter, are considered not versionable, historical information may be stored via the configuration information stored for the item that contains the module parameter or other non-versionable item. Another example of a non-versionable item is a step or transition.

In one embodiment, the check-out operation is limited to the extent that only a single check-out is permitted. In this manner, two or more users may not attempt to check-out the same item. In the event that a user attempts to check-out a previously checked out item, the VCAT system 98 generates a dialog window (not shown) to inform the user of the previous check-out, together with the identity of the user responsible for the prior check-out. Alternatively, the VCAT system 98 may include the functionality necessary to track concurrent modifications of each item.

In another embodiment, initiation of the check-out operation causes the VCAT system to analyze the selected item to determine which, if any, items in the version control database 102 may be affected by modifications to the selected item. Other items may, for instance, be affected if they utilize the selected item. The items that may be affected by modifications to the selected item can then be identified in a dialog window (not shown) generated by the VCAT system 98 to permit the user to select which, if any, of the items should also be checked out. A similar dialog window may be generated during the check-in procedure to alert the user of any checked out items that are referenced by, relied upon by, or otherwise associated with the selected item.

The option to initiate the check-out and check-in operations, or any other task executed within the VCAT system 98, may be provided within the user interface 94 in a number of ways. In a windows-type environment such as that described and shown above in connection with the configuration applications 96, a user may select from a top level menu bar 112 a "Tools" option, which results in the display of a plurality of

available tasks in accordance with known windowing techniques. The same or a similar menu may be generated within the user interface by "right-clicking" on an item in the main window 80 of the configuration application 96. In that case, the menu may be referred to as a "context menu."

FIGS. 6 and 7 show examples of a drop-down menu 114 and a context menu 116, respectively, that may be generated within the user interface 94 by the VCAT system 98. The drop-down menu 114 includes a plurality of task items that may be selected by the user via a pointing device or the like. A version control item 118 is shown as highlighted (after appropriate selection by the user) to generate a version control sub-menu 120 in accordance with known windowing techniques. The version control sub-menu 120, in turn, includes a plurality of task items that correspond with tasks that may be initiated and/or executed by the VCAT system 98. The context menu 116 similarly includes a plurality of task items that correspond with VCAT system tasks that can act upon the item selected within the main window 80. In either menu, commands that are not available for execution for whatever reason (*e.g.*, inapplicability or lack of authorization) may be displayed as disabled in a different font, style, etc. For example, in the sub-menu 120 of FIG. 6, the item selected presumably is yet to be checked-out and, accordingly, the check-in task cannot be initiated. Each of the tasks made available via the menus 116, 120 are described hereinbelow in connection with the operation of the VCAT system 98.

With reference now to FIG. 8, if the user selects the "Options" item in the context menu 116 (FIG. 7) or, alternatively, a "Preferences" item set forth in another menu (not shown), a version control preferences dialog 122 is generated by the VCAT system 98 for display on the user interface 94. The version control preferences dialog 122 enables the user to establish user preferences for the environment created by the VCAT system 98. As set forth in detail in Table 1 below, the version control preferences dialog 122 provides checkboxes for toggling between three respective options. The first option determines whether the user prefers to manually check items out. The second option in the version control options dialog 122 concerns whether an item will be automatically checked out when the user attempts to, or otherwise

initiates, the modification of an item. In the event that items are automatically checked-out, the VCAT system 98 becomes more transparent to the process designer utilizing the configuration applications 96. The third and final option concerns whether items will be automatically checked out and checked in when the user attempts to save an item after a session that provides an opportunity for modification.

It should be noted that the version control preferences dialog 122 may include other options to be elected or enabled by each user. For example, each user may be given the option of not providing comments during a check-in operation. Further details regarding the provision of such comments, however, will be provided in connection with Table 2 below.

TABLE 1 - Version Control Preferences Dialog

Name	Type	Min	Max	Default	Content
Manual Check-out	Radio Button	n/a	n/a	Selected	Determines if a dialog is displayed to prompt user to check-out an item for modification.
Automatic check-out	Radio Button	n/a	n/a	Not selected	Indicates if item will be automatically checked out.
Automatic Check-out and Check-in	Radio Button	n/a	n/a	Not selected	Indicates if item will be automatically checked out and checked in when modifying an item configuration.

The user may wish to select automatic check-out because, for instance, changes to one versionable item may affect and cause changes to one or more other versionable items. For example, the modification of an item such as a composite function block may affect one or more modules that use the composite function block. The VCAT system 98 preferably determines during each check-out operation which other versionable items need to be checked out in order to modify the configuration of

an item. The modification of these other versionable items may be referred to as "consequential changes." If the user has elected manual check-out, then the VCAT system 98 prompts the user to check these items out. If automatic check-out is enabled, the VCAT system 98 does not prompt the user and automatically checks out each of the other items for which consequential changes may occur.

The user may check-out and check-in items manually by utilizing the appropriate command offered via the version control sub-menu 120 or the context menu 116. Preferably, the VCAT system 98 then determines whether the selected item is a versionable item. At this time, the VCAT system 98 also determines whether the selected item has already been checked out. In the event that the selected item is already checked out, a message dialog (not shown) will alert the user of the fact and provide the user with an opportunity via, for instance, a button, to dismiss the message dialog and return to the previous state of the user interface 94.

Because the configuration of the process 10 is set forth in a hierarchal manner, the VCAT system 98 must allow for checking out items having subordinate items that are also versionable. Accordingly, during a check-out operation, the user is provided with the option of recursively checking out any such items. A similar option is provided in connection with the check-in operation. In one embodiment, if a recursive check-out or check-in is selected by the user, the VCAT system 98 generates a dialog window (not shown) that provides the user with a list of versionable, subordinate items that may be checked out (or checked in). The user may then select (or de-select) any one of the listed items to initiate a selective recursive operation.

When the VCAT system 98 is integrated with the configuration applications 96, the appearance of the items displayed within the main window 80 may be modified to denote that the item has been checked-out. In one embodiment, a checkmark overlays the icon associated with the item. It should be understood that a variety of other appearance schemes may be utilized. Checkmarks of different colors may also be utilized to denote specific users or when the item was checked out by the current user or a different user.

With reference now to FIG. 9, checked out items may be manually checked in using the appropriate commands made available in the version control sub-menu 120 and the context menu 116. A check-in dialog window 124 is preferably generated upon user initiation of the check-in procedure when the user has elected to proceed with a manual check-in environment. The check-in dialog window 124 includes a comment field 126 for accepting user-supplied commentary regarding the modification of the item. The commentary may, for instance, include an explanation of why the modifications were made. In addition to the comment field 126, the check-in dialog window 124 provides several other options to the user, each of which is described in detail below in Table 2. In particular, the user is provided with the options of designating that any subordinate items associated with the item to be checked in (that are also checked out) should be checked in, and electing that the item should remain checked out for further modifications. As set forth above, the check-in operation also stores data representative of the configuration in the version control database 102 as the latest configuration version. In this case, however, even though the item has been checked in, the opportunity to make further modifications to the configuration of the item remains persists because the item remains checked out.

The check-in dialog window 124 provides "OK" and "Cancel" buttons 128 and 130 for executing or canceling the check-in procedure, respectively, as well as a "Differences" button 132 that initiates a comparison of the current version of the item (as represented by data stored in the configuration database 100) and the modified version to be checked in. The comparison information is generated by the VCAT system 98 by accessing the configuration database 100 and the latest version in the version control database 102 and presented via the user interface 94. The generation and presentation of the "Differences" information will be discussed in greater detail hereinbelow.

TABLE 2 - Version Control Check-In Dialog

Name	Type	Min	Max	Default	Content
Recursive	Checkbox	n/a	n/a	Not checked	Checks in any subordinate items that are checked out by this user; preferably only applicable to certain levels of the configuration hierarchy.
Keep checked out	Checkbox	n/a	n/a	Not checked	Indicates whether the VCAT system should keep the item checked out.
Differences	Button	n/a	n/a	n/a	Difference between latest version (to be checked in) and the current version in the configuration database.
Comment	Edit field	n/a	n/a	Empty	User-supplied description of change.

Upon checking in an item, the VCAT system 98 may modify the appearance of the item to indicate that the item has been checked-in. This appearance modification may, for example, amount to the removal of a check-mark over the icon associated therewith. The VCAT system 98 may further include functionality that permits the user to request a status update for all of the items in the configuration database 100 to ensure that those items that are checked out are indicated as such via a checkmark or the like. Such functionality may be necessary, for instance, if two or more users are working on configuring the process at the same time, and one of the users has recently checked out an item shown on the user interface 94 of another user.

Selection of the "Cancel" button within the check-in dialog window 124 closes the Check-in dialog window without initiating a check-in operation.

The VCAT system 98, however, may also enable a user to initiate an "undo checkout" task that removes the check-mark over the icon and, if the item was modified, retrieves the data indicative of the latest version from the version control database 102, and imports the data into the configuration database 100. This procedure restores the configuration of the versionable item to the version that was current as of the time of the check-out operation. This task may be made available via a drop-down or context menu associated with the VCAT system 98.

The configuration applications 96 may provide additional ways to initiate the check-out/check-in operations. For example, the user may select an item in the Explorer system and access the "Properties" associated therewith via a right-click or similar pointer maneuver. A Properties window is then generated in accordance with standard windowing techniques that provides for editing of the contents of various "properties" fields displayed within the window. A module, for instance, may have a textual description associated therewith, or utilize certain parameters during execution within the process. The textual description and parameter values may be specified as "Properties" of the module. If the user modifies a property and selects an "OK" or "Apply" button, the VCAT system 98 prompts the user to check-out the item (if automatic check-out is not enabled), after which a check-in procedure is initiated as described hereinabove. Similar actions may be generated via utilization of an "Open" command made available in a command menu of the process configuration applications.

A check-out procedure may also be initiated by the VCAT system 98 when a user attempts to rename an item within the Explorer system. In accordance with standard windowing techniques, the user may select the name associated with an item to allow for editing thereof. Once the user is finished editing the name and attempts to enter the modification, the new name will only be accepted if the item is checked out. If the item is already checked out, the item is renamed in the configuration database 100, and data indicative of a new version of the item is stored in the version control database 102 when it is checked in. At that time, the VCAT system 98 may generate further data representative of commentary that describes the renaming

operation, and then associate such data with the data indicative of the new version of the item. If the item to be renamed is not yet checked out, the user is prompted to check-out the item via a dialog window (not shown) to that effect (if automatic check-out is not enabled).

As set forth above, data representative of each prior configuration of an item is stored in the version control database 102 together with data reflective of a version assigned thereto. The version is preferably identified by number, but may be indicated in any other manner. Having each prior configuration associated with a version number may assist the user during an analysis of the contents of the version control database 102, as well as when the process is in the run-time environment. To this end, the version number or identifier of an item is preferably included as an item parameter when the item is downloaded to the controller 12 and/or a batch executive routine. (The batch executive routine resides on the workstation 14 to manage the execution of a batch process and oversee the controller 12 in connection therewith.) In other words, the data downloaded from the workstations 14 when the process control system 10 is preparing to implement a process includes version identifying data for certain items in the configuration database 100 (*i.e.*, each module, phase, procedural element, etc.) that are involved in the download operation. The version information is then available in the run-time environment to a process operator via the controller 12 and/or the batch executive. Knowledge of the version information may assist in the communication between the process operators and the process designers working within the configuration applications 96.

It should be noted that downloading an item to the controller 12, or the run-time environment in general, involves one or more dialog windows (not shown) generated by the configuration applications 96 for the user interface 94 that guide the user through the download procedure. These dialog windows may correspond with the same windows generated by the configuration applications 96 when the VCAT system 98 is not integrated therewith. However, the VCAT system 98 may verify that all items to be downloaded are not checked out before allowing the download procedure to proceed. If any items are checked out, the VCAT system 98 may

generate one or more error dialog windows (not shown) that alert the user to the situation and identify the checked out items. Such error dialog windows may facilitate the check-in procedure by providing a button therein directed to initiating a check-in procedure of one or more checked out items identified in the dialog window.

With reference now to FIG. 10, the version identifying information may be made available to the process designers via a version audit report (hereinafter "audit report") generated by the VCAT system 98 in accordance with one embodiment of the present invention. In general, each change or modification to the process configuration is recorded in the version control database 102 as a specific version associated with the process. The data indicative of the version and the substance of the modification may be linked to data representative of the check-in date and time, the user who checked-in the item, and the reasons for the modification. It should be noted that the VCAT system 98 may also store data representative of the check-out date and time, as well as the identity of the user checking out the item.

The audit trail report may be provided to the user via an audit trail dialog window 140 generated by the VCAT system 98 for the user interface 94. More particularly, a "Show history" task may be initiated by the user by using one of the above-described techniques involving, for instance, the context menu 116, while the user is within one of the configuration applications 96. The audit trail of history may also be provided to the user via any other display device, such as a printer, or may be embodied in an electronic version of the report such that, for instance, data representative of the audit trail is delivered electronically over a network.

The audit trail dialog window 140 includes a table portion 142 having a header 144 that identifies a plurality of field names for presenting version information relating to a version number, a user name associated with the check-out, the data and time of the check-in, and a description of the modification or action. The version information is presented in lines (or records) beneath the field names as shown in FIG. 10. Each record is selectable via a pointer or otherwise to initiate a variety of operations. The functionality provided by the VCAT system 98 in connection with the audit trail dialog window is summarized below in Table 3.

TABLE 3 - Audit Trail Dialog Window

Name	Type	Min	Max	Default	Content
Version, User, Date, Action	List control	n/a	n/a	n/a	Each line corresponds with a history record for the item.
Close	Button	n/a	n/a	Enabled	Closes the dialog.
Rollback	Button	n/a	n/a	Enabled	Rollback to the selected version.
Differences	Button	n/a	n/a	Enabled	Compares two selected versions or selected version with current version.
Details	Button	n/a	n/a	Enabled	Displays comments of selected version.
View	Button	n/a	n/a	Enabled	Displays the exportable item in text or graphical format.

As set forth in Table 3, the user may view the details of a version of the item by selecting the line of the table portion 142 corresponding with that version and then selecting a View button 143. The data stored in the version control database 102 is then accessed to present the configuration information for the selected version of the item in either a textual or graphical format, or both, based on the nature of the item. Certain items, for instance, may only constitute text and, thus, may not be viewed in the graphical format. For example, versionable items such as a workstation or I/O device only contain parameters that can be modified. The parameters are set forth in text and, thus, no graphical view would be available for such items. Some items, however, may be viewed in both the graphical format and the textual format and generally include items such as modules, composite function blocks, or other items

based on a sequential flow chart algorithm. In such cases, the user may be prompted to elect one of the formats or, alternatively, both formats may be presented.

An example of an item shown in the textual format is shown in FIG. 11, which is a screen display generated by the VCAT system 98 for the user interface 94. The screen display includes a text view window 146 having a frame 148 for the textual information associated with the item. The text view window 146 provides the user with several viewing options or operations which will be described hereinbelow.

FIG. 12 is an example of a screen display of an item shown in the graphical format. Like the screen display for the textual format, the graphical format is presented via a graphical view window 150 having a frame 152 for the graphical information associated with the item.

Returning to FIG. 11 and Table 3, the user may also analyze any comments recorded at the time of check-in for a particular version by selecting a Details button 154 after highlighting the desired version. The comments are preferably stored via the check-in dialog window 124, but other schemes may be utilized to associate data representative of comments with a particular version. The audit trail dialog window 140 further allows the user to select two versions of the item by using a pointer (or other selection mechanism) to highlight the corresponding two records in the table portion 142. Highlighting two records enables the user to initiate a Differences operation by selecting a Differences button 156, which, in turn, causes the VCAT system 98 to generate a Differences window dialog that provides the configuration information for each selected version of the item in a format that permits the user to compare the two versions. The details of the Differences operation will described in greater detail hereinbelow. If only a single record has been highlighted, and the Differences button 156 is selected, the VCAT system 98 will generate the Differences window such that the selected version is compared with the current version of the item stored in the configuration database 100.

The audit trail dialog window 140 also generally enables the user to implement one of the prior versions of the configuration of the item. More particularly, the VCAT system 98 may be directed by the user to initiate a routine that accesses the

version control database 102 to retrieve the data representative of a prior configuration version in order to modify the data stored in the configuration database 100 in connection with the selected item. The modification of the configuration database 100 may then occur by importing the data from the version control database 102 into the configuration database 100. To implement this "rollback" to a prior configuration version of an item, and in accordance with one embodiment, the audit trail dialog window 140 includes a Rollback button 158 that may be selected by the user once a record in the table portion 142 has been selected.

In one embodiment, the VCAT system 98 provides the user with the option to rollback to a configuration that was downloaded to the controller 12 for utilization within the run-time environment. Such a rollback operation may be initiated by selection of a task item such as "Recover download" from a drop-down or context menu, and include the generation by the VCAT system 98 of a specialized audit trail dialog window (not shown) that only lists those versions that were downloaded. As will be described in greater detail hereinbelow, the version control database 102 preferably includes data indicative of whether a version was downloaded for use in the run-time environment. A version may then be selected by the user, the configuration information for which is then retrieved from the version control database 102 by the VCAT system 98 and stored as the latest configuration version.

Preferably, the VCAT system 98 determines whether the configuration data import operation is successful by checking for other items in the configuration hierarchy and process that depend upon the item. For example, when a configuration version of a module is being restored, the configuration version may require a certain subordinate item (*e.g.*, composite function) block to exist. The module is then said to have a "dependency" in the sense that the success of the rollback operation is dependent upon the continued existence of that subordinate item. A module may have numerous different types of dependencies, such as embedded function blocks, linked function blocks, enumeration sets, alarm types, other modules, plant areas, and controllers. Accordingly, in one embodiment, the VCAT system 98 checks each

subordinate item of the item being subjected to a rollback to determine whether any versionable items have been deleted.

In a preferred embodiment, the data imported into the configuration database is checked into the version control database 102 as well as the latest configuration version of the item. The check-in occurs concurrently with the modification of the configuration database 100, and preferably includes the storage of data representative of a comment indicating that the user reverted back to a prior configuration version of the item. Alternatively, the VCAT system 98 may generate a dialog window to provide the user with the option of entering a customized comment, entering a default comment, or no comment at all. If the item was checked out before the rollback task was initiated, the VCAT system 98 may, but need not, keep the item checked out after the rollback task is completed.

Lastly, the audit trail dialog window 140 includes a button 160 for dismissing the dialog window 140 and returning the user interface 94 to its previous state.

As shown in FIG. 10, one of the actions (other than a check-in operation) recorded in the audit trail report is a label setting operation for the entire process configuration. With reference now to FIG. 13, the user may initiate a Set Label operation by selecting a versionable item within, for instance, one of the modules designed using the configuration applications, and selecting the Set Label task via either a drop-down or context menu generated by the VCAT system 98. In response, the VCAT system 98 generates a Set Label dialog window 162 that includes a label field 164 and a comment field 166. The user is then prompted to enter a label to be applied to the selected item (and subordinate items) in the version control database 102. The latest configuration version of each item stored in the version control database 102 is then assigned the user-entered label. The audit trail data stored in the version control database 102 is also updated to reflect the assignment of the label, which is preferably applied to the latest configuration version. More particularly, the version action in the audit trail configuration history data will reflect that the version corresponds with a newly assigned label having the entered name, and the details

associated therewith will correspond to the information entered by the user in the comment field 166 of the Set Label dialog window 162.

Setting a label may be advantageous for a process designer in the sense that the process designer may manually note via a label, such as "DOWNLOADED TO CONTROLLER1," which configuration versions of various items were downloaded to the controller 12 for implementation in the process. Preferably, however, the VCAT system 98 automatically assigns a label for each downloaded item to flag the configuration version as having been downloaded to the run-time device. Data representative of further information such as the date and time of the download and the items that were downloaded concurrently may also be stored in connection with the label.

The VCAT system 98 may then accordingly provide the capability to revert back the configuration of the entire process, or some portion thereof, to the version associated with a label set via the above-described procedure for each of the affected items in the version control database 102. In one embodiment, this capability applies only to the root (or top level) item in the configuration hierarchy developed using the configuration applications 96. In such case, a "Revert System to Label" task may be initiated when the user has selected the root item and proceeds to select the appropriate task item from a drop-down menu or a context menu similar to the menu shown in FIG. 7. A warning dialog (not shown) may be generated by the VCAT system 98 prompting the user for verification of intent to revert the system back to a label. If the user indicates that a "Revert System to Label" task is desired, the user may then be prompted for the name of the label to which the system will be reverted back. Alternatively, the VCAT system 98 may automatically revert the system back to the last label set in the version control database 102. In any event, once the Revert System to Label task is executed, the VCAT system 98 accesses the version control database 102 to retrieve the data representative of the configuration version associated with the label for each item, and imports the retrieved data into the configuration database 100. The VCAT system 98 then modifies the version control database 102 to reflect the reversion by storing data indicative of a new configuration version for

each item as well as data associated therewith that reflects the reversion back to the label. In one embodiment, this data may be associated with the configuration version by storing data representative of a comment that indicates the reversion back to the label.

In accordance with one embodiment, the VCAT system 98 preferably still stores data in the version control database 102 that is representative of an item that was deleted or otherwise removed from the configuration database 100. In such a case, the version control database 102 may be relied upon when and if the user decides to revert back to a process configuration utilizing the deleted item. Referring now to FIG. 14, the VCAT system 98 may also enable a user to purge items that have been deleted from the configuration database 100. To that end, the VCAT system 98 generates a Recover/Purge dialog window 168 having a list of such deleted items. The Recover/Purge dialog window 168 may be generated upon selection of a Recover/Purge task item in either a drop-down or context menu. Such a task item may be enabled or other made available to the user after selection of an item within the configuration applications 96. If the selected item has one or more subordinate items that have been deleted from the configuration database 100, the user may initiate the recover operation via the dialog window 168 to add data representative of one or more of the deleted items to the configuration database 100.

The Recover/Purge dialog window 168 includes a Recover button 170 for restoring one or more deleted items in the list that have been selected by the user. When restoring a deleted item, the VCAT system 98 may access the version control database 102 to retrieve the latest configuration version of the deleted item, and then import the data associated therewith into the configuration database 100. Alternatively, the VCAT system 98 prompts the user for selection of a prior configuration version to the extent multiple versions are stored in the version control database 102.

Selection of a purge button 172 provided in the dialog window 168, on the other hand, results in the removal of any data associated with the item from the

version control database 102. Further details regarding the Recover/Purge dialog window 168 are provided below in Table 4.

TABLE 4 - Recover/Purge Dialog Window

Name	Type	Min	Max	Default	Content
Items	Check list boxes	n/a	n/a	List of subordinate items deleted based upon a selected parent	List all subordinate items deleted based upon a selected parent.
Recover	Button	n/a	n/a	n/a	Item is restored to the configuration database by retrieving the data from the version control database and importing the item into the configuration database.
Purge	Button	n/a	n/a	n/a	Item and all audit trail history data are removed from the version control database permanently.

In accordance with one embodiment of the invention, the VCAT system 98 supports version control and audit trail functionality for items that are not created, developed, or managed by the configuration applications 96. For example, a user may create a document in a word processing program that describes the functionality of a particular item. It would be desirable to store and track any editing of the document as the configuration and, hence, the description. are modified. To this end, the main control window 80 generated by the Explorer system for the user interface 94 is utilized to designate a new version control item. More particularly, the user first

expands a "User Workspace" object (*i.e.*, an object in the Explorer system). A task item directed to creating a new folder within the hierarchy frame 82 is then selected by the user, and a new folder is created and named in accordance with known windowing techniques. Similarly, a task item directed to creating a new item is selected, and the user is then provided with the opportunity to browse for a document or file to insert as a new item in the recently created folder. Once the document or file is inserted into the folder, it is then added to the version control database 102 as another item for which versions and version-related information can be stored. Subsequently, any version of the file or document may be retrieved from the version control database 102 using a Get task item made available to the user via a drop-down or context menu. The Get task provides the user with a dialog window (not shown) that prompts for a desired storage location for the retrieved file, such as a floppy disk. The native application utilized to create the file or document (*e.g.*, a word processing application) may then be used to access the contents of the file or document from the designated storage location.

With reference now to FIG. 15, the VCAT system 98 preferably generates a message dialog window 174 for display of version control feedback information to the user via the user interface 94. The message dialog window 174 includes a text box 176 in which textual descriptions of each version control operation is documented as it occurs. Generally, the message dialog window 174 permits the user to scroll through a history of the operations that have occurred since a Clear button 178 was last selected by the user. An Abort button 180 is also provided to enable a user to abort a version control operation at the next opportunity. For example, if the user has initiated a rollback operation, execution of the operation by the VCAT system 98 may be ceased via selection of the Abort button 180 unless, for instance, execution of the operation has already been completed. The message dialog window 174 may automatically minimize in accordance with known windowing techniques if version control operations have not occurred for a certain period of time. The message dialog window 174 is then automatically re-displayed on the user interface 94 upon initiation of the next version control operation.

The manner in which the VCAT system 98 provides the user with a textual view of the configuration of an item via the textual view dialog window 146 will now be described in connection with FIG. 11. The VCAT system 98 may, but need not, be capable of displaying the configuration information of each item in the version control database 102 in a textual format. As explained above, some items, by their nature, may have configuration information that also may be displayed graphically.

The format in which the configuration information for an item is set forth is not pertinent to the practice of the present invention. However, the VCAT system 98 preferably utilizes a textual format that minimizes the amount of horizontal scrolling necessary for user viewing of both a single configuration version of an item as well as when comparing two versions. Furthermore, it is preferred that key words and labels be utilized to identify attributes such as object type and properties. Examples of item-type identifying labels that may be utilized include "PARAMETER," "FUNCTION_BLOCK," and "CONTROLLER." Examples of property labels are "NAME" and "DESCRIPTION." The key words and labels may then be translated to the appropriate language. For example, a Japanese version of the user interface 94 would display the key words in Japanese. Conditional and other logic statements in the language may also be translated into an easy-to-read format for display in the textual view.

In one embodiment, the textual format for each item includes delimiting language such as BEGIN and END. For example, as shown in FIG. 11, configuration version number 13 of a module has a parameter delimited by the following BEGIN and END statements:

```
BEGIN Parameter "ABS_PRESS_CF"
END Parameter "ABS_PRESS_CF"
```

In another embodiment, the name of the item may not be identified in conjunction with a label, such as in the following example:

```
BEGIN FUNCTION_BLOCK
    NAME = "AI1"
    DEFINITION = "AI"
    DESCRIPTION = "Analog Input"
    LEFT = 200
```

TOP = 200
HEIGHT = 125
WIDTH = 400
END FUNCTION_BLOCK

The textual view dialog window 146 includes a find button 200, a find down button 202, and a find up button 204. The buttons 200, 202, and 204 generally allow the user to navigate the textual information set forth in the frame 148 in accordance with a character string selected therein with a pointer or the like in accordance with known windowing techniques. To this end, initiating the find operation by selecting the find button 200 may cause the VCAT system 98 to generate a find dialog window (not shown) to facilitate the user's navigation of the frame 148. A character string may then be entered into a field and, the first instance of the string may be found by, for example, selecting an "OK" button (not shown) in the find dialog window. The user may subsequently use the buttons 202, 204 to find instances of the string in either the down or up directions, respectively. Further information with regard to these operations and others provided via the textual view dialog window 146 is set forth below in Table 5.

TABLE 5 - Textual View Dialog Window

Name	Type	Min	Max	Default	Content
Find	Button	n/a	n/a	Enabled	Allows user to enter a search string.
Find down	Button	n/a	n/a	Disabled	Navigates to next instance of the search string
Find up	Button	n/a	n/a	Disabled	Navigates to previous instance of the search string

At this point, it would be instructive to describe in greater detail the manner in which the textual information set forth in the dialog window 146 is generated by the VCAT system 98 from the data representative thereof stored in the version control

database 102. It should first be understood that the manner in which the version control data is stored is not critical to practice of certain aspects of the present invention. In one embodiment of the present invention, however, the version control data is preferably stored in the version control database 102 in a file-based format. In an alternative embodiment, the data is stored in an object-oriented fashion such that the version control database comprises an object-oriented database.

The version control database 102 is preferably administered using the DeltaV™ Database Administrator application available from Fisher-Rosemont, Inc. modified to the extent necessary to handle both of the databases 100, 102. Alternatively, the Microsoft SQL Server® Enterprise Manager may be utilized for this purpose.

To generate the textual information representative of the configuration version, the VCAT system 98 executes a routine that generally accesses the version control database 102 to export the pertinent data in a manner that can be translated into either a text- or graphical-based format. To this end, during a check-in operation, the VCAT system 98 stores a text-based representation of the version control data in a file in accordance with a markup language, such as XML (Extensible Markup Language). The text contained in the XML document that is generated at this point may be serialized into a single character string that is stored in the version control database 102. More particularly, in one embodiment, each versionable item may have a database record corresponding to each configuration version. In that case, each configuration version record has a field dedicated to having a single character string of XML text stored therein that represents the version control data associated with the configuration version. Preferably, these configuration version records make up one table of a plurality of tables in the version control database 102, which, in this case, is a relational database. The relational database may include other tables directed to storing the following: (1) whether each versionable item is deleted, the current version identifier, whether the item is currently checked out and, if applicable, to whom; and, (2) the audit trail information for each versionable item.

In a preferred embodiment, all of the configuration data necessary to completely define a configuration of an item is separately stored for each version. Alternatively, the configuration data may be stored in manner that only defines the differences between versions, in which case the data associated with multiple database fields would have to be accessed to develop the XML document for a particular version

The version control data is thereby set forth in a structured manner that can be easily manipulated and parsed when the version control database 102 is later accessed. To develop the configuration information shown in the example of FIG. 11, the VCAT system 98 interprets the XML representation of the version control data to provide the configuration information in an easy-to-read textual format. The manner in which the XML-formatted data is processed should be readily appreciated by those skilled in the art, but will now be briefly described in greater detail.

At the time of the filing of the instant application, XML is a markup language well known to those skilled in the art and believed to be in the midst of the standardization as XML 1.0 by the World Wide Web Consortium (www.w3.org). While other data schemes may be utilized, the use of any one of a number of markup languages is preferred to facilitate the generation of both textual and graphical representations of the configuration of the item.

Generally speaking, a document set forth in XML has a logical structure composed of declarations, comments, and processing instructions, as well as nodes that start with a single root (or document) element. For example, a versionable item may have an initial or root "MODULE" element to represent a portion of a control strategy. This MODULE element may contain an attribute node called NAME, which has a text node containing the name of the module. Additionally, the module may contain element nodes to describe the module. Such elements may include, for example, ALGORITHM_TYPE, DESCRIPTION, EXECUTION SPEED, AND MODULE_TYPE. Each of these elements may have a textual node that maintains a value representative of that particular portion of the configuration. The MODULE element node may further include element nodes that contain additional elements,

thereby creating a hierarchy. For example, a MODULE may include a STEP, and the STEP may include one or more ACTION elements. The ACTION elements, in turn, include elements such as NAME, DESCRIPTION, ACTION_TYPE, and the like.

Thus, the XML format sets forth the version control data with the aid of tags in a manner similar to that found in HyperText Markup Language (HTML) documents. In XML documents, however, the tags may be customized to the types of data being stored, as described above. For example, the function block for which a textual representation was provided hereinabove is set forth below in XML format:

```
<Function_Block>
        <Name>AI1</Name>
        <Definition>AI</Definition>
        <Description>Analog Input</Description>
        <Left>200</Left>
        <Top>200</Top>
        <Height>125</Height>
        <Width>400</Width>
</Function_Block>
```

The data stored in an XML document is accessed in accordance with an object model that provides for parsing the document to create a data tree structure having a plurality of nodes associated with the version control data. Practice of the present invention may utilize any one of several different object models, such as the Document Object Model as applied to XML.

Thus, the textual information provided in the textual view dialog window 146 via the user interface 94 is generated by loading the XML text in the database field to create the above-described object nodes for the configuration version. The document object model may then be used to traverse the data structure established by the object nodes in order to extract the configuration information and generate the textual format shown in FIG. 11 for the selected version and item.

The graphical view dialog window 150 of FIG. 12 is also generated from the above-described XML-based routine. As will be readily recognized to those skilled in the art, the same XML document utilized to generate the textual information may be able to support a graphical representation, given the necessary objects in the tree

structure, together with the knowledge of how certain objects are to be displayed. With a limited number of object types available, such knowledge is provided by the VCAT system 98. For instance, when the XML document indicates that an item is a function block, the VCAT system 98 may initiate a routine for a generic drawing function block and, in so doing, refer again to the data provided in the XML document to draw the function block in accordance with other parameters.

It should be understood that not every item may be displayed graphically. However, items that may be displayed graphically include, but are not limited to, those that are generally defined by a function block or sequential flow chart algorithm.

The graphical view dialog window 150 that results from a user's selection of a configuration version in the audit trail dialog window 140 provides a user with various options to facilitate review of the configuration. More particularly, the dialog window 150 includes a text view button 206 that allows the user to display the configuration information displayed in the frame 152 in a textual format in accordance with the above-described manner. The text view button 206 may also be directed to a particular subordinate item displayed in the frame 152. More particularly, after a user has selected the subordinate item, the text view button 206 may be selected to view the configuration information of the subordinate item in a textual format. Alternatively, the subordinate item may be selected using a right-click or the like to generate a context menu 207 that provides the text view task as an item to be selected.

The dialog window 150 further includes a Display Parent button 208 and a Drill Down button 210. The Drill Down button 210 provides the user with the option to display (to the extent possible) the configuration information for a subordinate item of the originally selected item in a graphical format. Once an item (such as a composite function block 212) displayed in the frame 152 is selected by the user, the Drill Down button 210 is enabled for selection by a user with a pointer or the like. The button 210 may then be selected to "drill" into the subordinate item to display graphical configuration information related thereto. The Drill Down task may also be initiated using the context menu 207.

The Display Parent button 208 provides the user with the option to return to a graphical view of the parent item by removing the view of the subordinate item. Alternatively, the subordinate item may be displayed in a separate dialog window such that the display parent operation causes the VCAT system 98 to close the dialog window associated with the subordinate item and thereby return to the graphical view of the parent item.

In an alternative embodiment, if no graphical view is available for a subordinate item (because, for instance, it does not constitute a composite function block or a module), then a textual view dialog window will be generated. It shall be understood that eventually the act of drilling into subordinate items within a graphical view dialog window will result in a textual view dialog window.

It shall be appreciated that utilizing an XML document generated on-the-fly as an intermediary between the version control database 102 and the generation of the user interface 94 provides an efficient, flexible approach to the presentation of configuration information. More importantly, however, generating an XML document for each configuration version also provides for the rapid and efficient comparison of two versions of a selected item.

Comparing versions of an item is useful in the general context of version control to determine what is commonly referred to as "Differences" between versions. With respect to the process control system 10 set forth hereinabove, it would be beneficial to display the differences between two versions of an item both textually and graphically. In a preferred embodiment, the VCAT system 98 generates Differences dialog windows for both textual and graphical presentations of the differences between two versions using the above-described, XML-based processing of the data stored in the version control database 102. Because the tree structure of an XML document is easily parsed in accordance with the object model, two versions of an item may be easily compared by parsing the corresponding two XML documents and comparing the parsed data object-by-object.

With reference now to FIG. 16, a textual differences dialog window 220 includes a first frame 222 and a second frame 224 for displaying configuration

information for a first configuration version and a second configuration version, respectively. Both horizontal and vertical scrolling of the textual information displayed in each frame 222, 224 is provided for using known windowing techniques. The first frame 222 may set forth the configuration information for an older version (*e.g.*, Version number 13), while the second frame may set forth the configuration information for a more recent version (*e.g.*, Version 14). It should be noted, however, that the versions need not be consecutively numbered, inasmuch as any two versions may be compared by the VCAT system 98. As the older version, the first frame 222 may include one or more lines of text that were deleted as a result of a check-out/check-in modification of the configuration. Preferably, the text associated with such deleted lines is shown in a different color (*e.g.*, blue), font, or style than the remaining text. To denote that blue-colored text, for instance, is representative of a deleted line, a button 226 may include the text "Deleted lines" in a blue color. Similarly, the frame 224 may include one or more lines of text that were inserted as a result of a modification to the configuration of the item. Such lines of text may, for instance, be set forth in a green color, as well as a button 228 having the green-colored text "Inserted Text." Lastly, the content of one or more lines of text may have been changed (but not deleted entirely) between versions. Such lines may be set forth in a red color in both of the frames 222, 224, together with a button 230 having the red-colored text "Changed Text." It should be understood that any color or style scheme might be utilized to differentiate the above-identified types of differences and to set the changed text apart from the text common to both versions.

The textual differences dialog window 220 also includes a find button 232, a find down 234, a find up button 236, a Down button 238, and an Up button 240 that provide the same navigational functionality described hereinabove in connection with the textual view dialog window 146. These navigation tools preferably apply to both of the frames 222, 224, such that any vertical scrolling operation initiated by the user results in the scrolling of both frames. In an alternative embodiment, these navigation tools (as well as other standard scrolling techniques in a windows environment) may

be applied to one of the frames 222, 224 based upon, for instance, which frame has been selected by a user.

The textual information for the versions being compared need not be set forth in a side-by-side manner. For instance, the textual differences dialog window may alternatively include a single frame wherein the common text, changed text, inserted lines, and deleted lines are shown and differentiated via a similar color or style scheme. Differences between the two versions may also be shown in that context by using redlining, underlining, demarcating punctuation, and the like.

A graphical differences dialog window 250 is shown in FIG. 17. The graphical differences dialog window 250 is based on a single frame window similar to the graphical view dialog window 150, but may alternatively use a two frame, side-by-side window to juxtapose the versions. To differentiate between common objects, deleted objects, added objects, and changed objects, a color scheme may again be employed. The colors may be applied to the objects as a background color, a border, a matting, an outline, etc. Buttons 252, 254, and 256 are set forth as a key to the color scheme in the same manner as set forth hereinabove in connection with the textual differences dialog window 220. For example, a step item S1 and a transition item T1 are shown with a colored border (*e.g.*, red) to indicate that the items have been modified. A further step item S2 and further transition item T2 are shown with a different colored border (*e.g.*, blue) to indicate that the items have been deleted or removed.

It should be noted that the color scheme may also be applied to the lines interconnecting the items.

Exemplary changes to an item that are denoted by a colored outline or frame include adding an action to a step and changing the expression of a transition. Changes to the execution order of the function blocks may be denoted by outlining, framing, etc. only the portion of the object dedicated to displaying the execution order. This portion may, for instance, be located at the bottom of an object. In the event that an object was renamed between versions, the color scheme may be employed to show the two versions of the object as deleted and added. Alternatively,

the VCAT system 98 may provide the user with the option of only displaying substantive differences between versions, such that cosmetic changes are not displayed.

The graphical differences dialog window 250 also includes a Display text button 258 that provides the same functionality as the button 206 with respect to both the selected item and any subordinate item.

If a subordinate item is shown as having been modified, the VCAT system 98 provides the user with the capability of drilling into the item (through appropriate selection thereof) to generate either a textual differences dialog window or another graphical differences dialog window for that subordinate item. In one embodiment, drilling down is only available for those items for which a graphical dialog window may be generated. In general, however, which type of dialog window will be generated depends on the type of item as explained hereinabove. Drilling into a subordinate item may also be initiated by selection thereof followed by selection of a drill down button 260. A return to the previous graphical differences dialog window 250 may be accomplished by selection of a Display Parent button 262 that operates in the same manner as the button 208.

When an item that has been removed is at the same location in the graphical differences dialog window as an item that has been added, one of the items may be obscured. Additionally, when a comments is changed, the old comment will not be visible in the graphical view. To allow the user to see the hidden item or comment, the VCAT system 98 may provide a mechanism (via a menu item or selection sequence using a pointer or the like) to toggle between which item or comment is shown (*i.e.*, on top) and which item is obscured from view.

In a preferred embodiment, the above-described textual view and graphical view dialog windows, including those directed to viewing differences, include a button or task item sequence that permits a user to toggle between textual and graphical displays. The VCAT system 98 would preferably only include such functionality on dialog windows for items that can be displayed in both formats.

It should be understood that the VCAT system 98 may impose a security check on the version control environment. For example, for an operation (*e.g.*, rollback) that requires a certain level or type of authorization, the VCAT system 98 may determine before executing the operation whether the user is authorized to initiate the operation.

When a new user is added to the VCAT system 98, the VCAT system 98 may generate a new user dialog window (not shown) having a plurality of checkboxes corresponding with a plurality of VCAT operations that may be selected or not selected to provide the user with a desired amount of authorization.

It should be further understood that the VCAT system 98 may provide a user having appropriate authorization with the option of enabling or disabling the VCAT system 98. Once disabled, the VCAT system 98 would allow the configuration applications 96 to operate without imposing the check-out/check-in and other procedures. When re-enabled, the VCAT system 98 preferably performs a synchronization routine that accesses the configuration database 100 to compare the current configuration data with the latest configuration data stored in the version control database 102. For each item that encountered a configuration modification during the period that the VCAT system 98 was disabled, a new configuration version is added, and data representative of the current version in the configuration database 100 is stored in association therewith. New items may also be created in the version control database 102 to the extent necessary. Furthermore, a label may be assigned to all items in the version control database 102 to denote that the VCAT system 102 has been re-enabled.

The synchronization routine may also be executed at any time. In certain instances, it should be appreciated that the VCAT system 98 may have to initiate one or more undo check-out operations and/or check-out operations to the extent necessary to modify the version control database 102.

The VCAT system 98 may also include a database backup/restore routine similar to the routine utilized to backup and restore the configuration database 100 administered by the Explorer system. Other common database utilities, such as a

"Clean Database" routine directed to repairing the data storage structure of the version control database 102, may also be included as part of the VCAT system 98.

The user interface 94 is preferably not the only output device by which version control information is provided to the users and operators of the VCAT system 98 and configuration applications 96. More particularly, one or more routines may be implemented by the VCAT system 98 to support the generation of reports for delivery to a printer or other display device. For example, the VCAT system 98 may provide the user with the ability to select task items directed to generating reports that present version control information related to which items are checked out, the audit trail of a particular item, the items checked out by a particular user, any deleted (but not purged) items, and a list of check-outs by date or some other parameter for which data may be stored in the version control database 102. Such information may be provided via any display device, whether located locally or remotely, and may be delivered in a format or in accordance with any protocol.

To facilitate the generation of such reports, the VCAT system 98 preferably includes a query system. The query system generally permits the user to specify search criteria directed to any number of subjects, such as modifications or actions by a particular user, modifications or version control events that occurred during a specified time frame, modifications or version control events that occurred within a specific version or label, and modifications or version control events that relate to a specific item or an area of items. After the user has entered the one or more criteria for a query, the VCAT system 98 will initiate a search routine that accesses the version control database 102 and analyzes the contents thereof. In a preferred embodiment, the VCAT system 98 executes the search routine in a background manner, such that other version control operations may be initiated and executed concurrently therewith.

The query system may generate one or dialog windows for display via the user interface 94. Examples of two such dialog windows are shown in FIGS. 18 and 19. A history report options dialog window 280 may be generated after selection of an item followed by selection of a task item directed thereto via either a drop-down menu

or context menu. The dialog window 280 provides checkboxes directed to whether the history report to be generated will include version control information concerning labels and subordinate items. A user field 282 is also provided to allow the history report to be directed to the modifications or version control events that were initiated by a specified user. Lastly, a "From date" field 284 and a "To date" field 286 may be used to specify a time period for the history report. To facilitate the entry of the starting and ending dates for the time period, a pop-up calendar or exemplary dates may be provided in a window generated for each field 284, 286 in accordance with known windowing techniques.

With reference to FIG. 19, the query system may generate a general search dialog window 290 that includes a status search portion 292 and a search area portion 294 for identifying certain search criteria. The status search portion 292 includes checkboxes for instructing the VCAT system 98 to search for all checked out items or items checked out to a certain user. The user may be specified via a user field 296 having the capability of suggesting user names via a drop down window. The search area portion 294 includes several checkboxes that permit the user to search the version control information relating to only the selected item, the selected item and all subordinate items, or all items in the version control database 102.

The screen displays of Figs. 5-19, as well as other screens, may be created and modified using a windows-type format with standard windows-type commands, although any other format could be used as well. The format of these screen displays may be changed dramatically, for instance, in the event the VCAT system 98 is used in conjunction with configuration applications other than the Explorer system or the other applications specified hereinabove.

Although the configuration applications 96 and the VCAT system 98 described herein are preferably implemented in one or more software routines, they may be implemented in a routine embodied in hardware, firmware, etc., and may be implemented by any other processor associated with the process control system 10. Thus, each of the operations and procedures described hereinabove may be implemented in a standard multi-purpose CPU or on specifically designed hardware

or firmware as desired. When implemented in software, the software routine may be stored in any computer readable memory such as on a magnetic disk, a laser disk, or other storage medium, in a RAM or ROM of a computer or processor, etc. Likewise, this software may be delivered to a user or a process control system via any known or desired delivery method including, for example, on a computer readable disk or other transportable computer storage mechanism or over a communication channel such as a telephone line, the internet, etc. (which are viewed as being the same as or interchangeable with providing such software via a transportable storage medium).

Thus, while the present invention has been described with reference to specific examples, which are intended to be illustrative only and not to be limiting of the invention, it will be apparent to those of ordinary skill in the art that changes, additions or deletions may be made to the disclosed embodiments without departing from the spirit and scope of the invention.

4 Brief Description of the Drawings

FIG. 1 is a block diagram of a process control system having a plurality of workstations and a controller for directing and configuring a process utilizing a process configuration system implemented on the plurality of workstations;

FIG. 2 is an exemplary screen display generated by the process configuration system to establish a user interface therefor via one of the workstations of FIG. 1;

FIG. 3 is a further exemplary screen display generated by the process configuration system to establish another user interface therefor via one of the workstations of FIG. 1;

FIG. 4 is a block diagram of the process configuration system integrated with a version control and audit trail system in accordance with one embodiment of the present invention; and

FIGS. 5-19 are screen displays generated by the version control and audit trail system of FIG. 4 to establish a user interface for the input and output of information relating to the configuration of multiple versions of the process in accordance with numerous embodiments of the present invention.

【Ｒｅｆｅｒｅｎｃｅ Ｎｕｍｅｒａｌｓ】

10 Process control system

12 Process controller

14 Workstation

18, 24 Processor

20, 26 Memory

96 Configuration application

98 Audit trail system (VCAT system)

100 Configuration database

102 Version control database

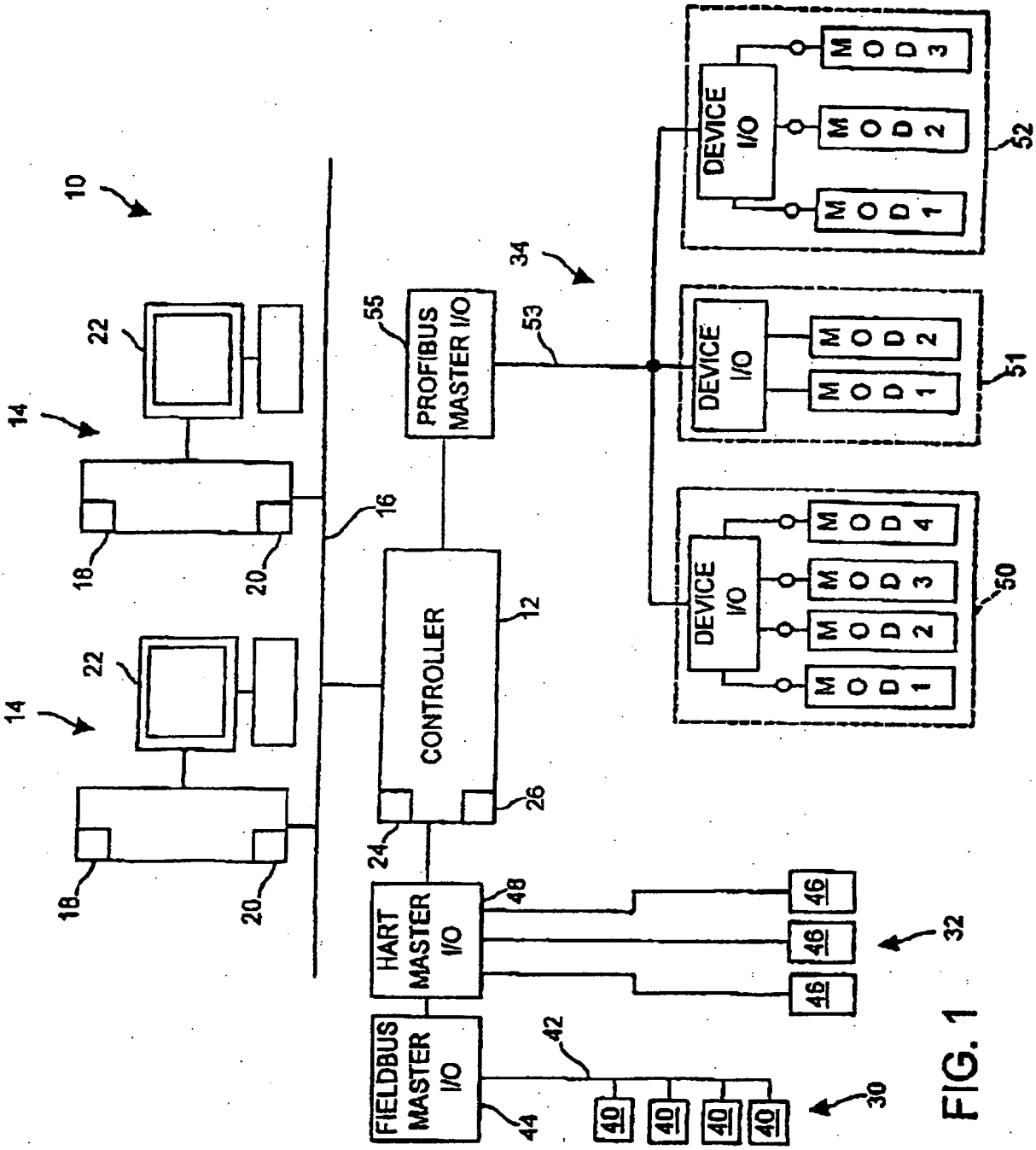

10
14
22
18
20
16
12
CONTROLLER
24
26
PROFIBUS MASTER I/O
55
53
34
DEVICE I/O
MOD 3
MOD 2
MOD 1
52
DEVICE I/O
MOD 2
MOD 1
51
DEVICE I/O
MOD 4
MOD 3
MOD 2
MOD 1
50
HART MASTER I/O
48
46
32
FIELDBUS MASTER I/O
42
44
40
30

FIG. 1

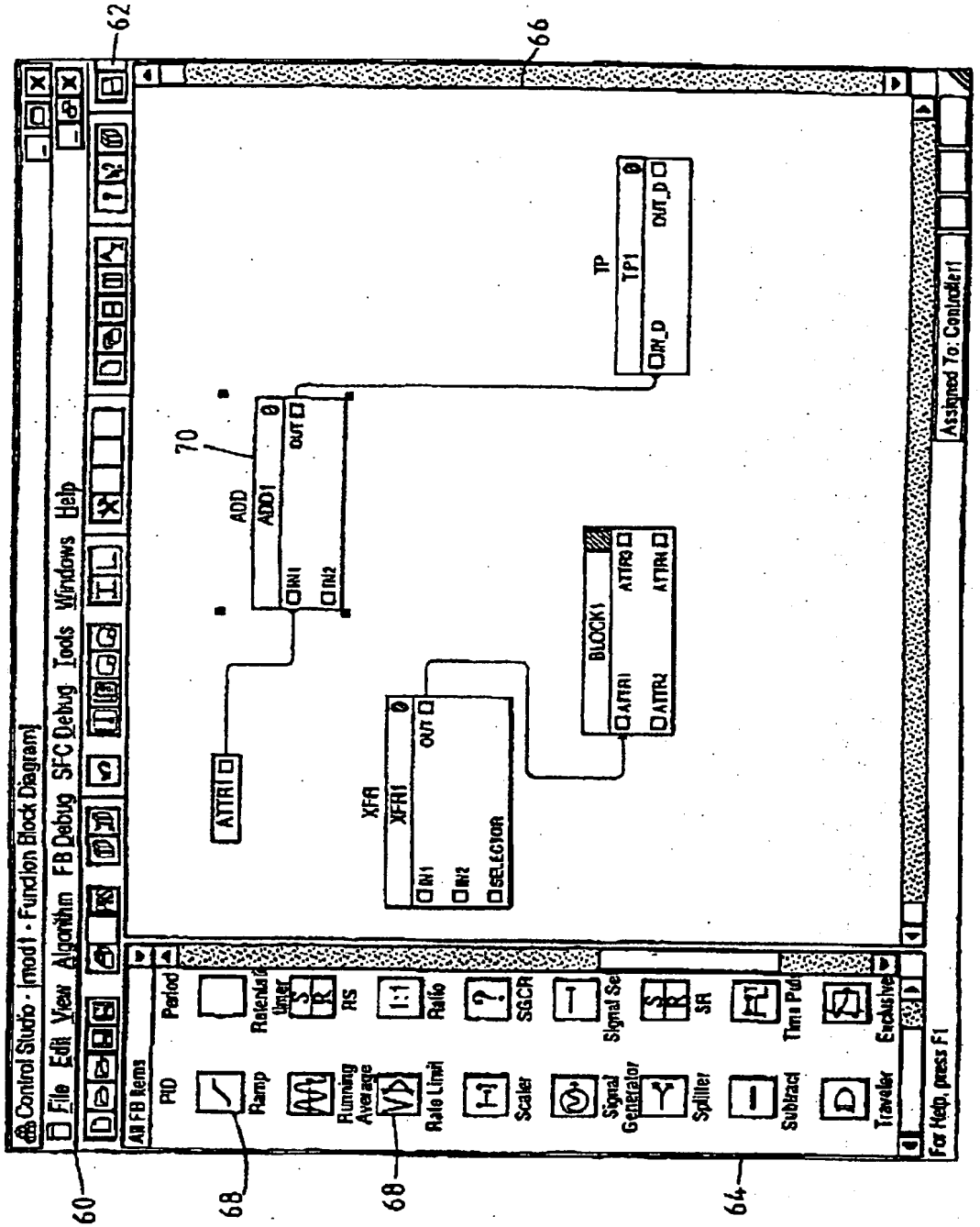

Control Studio - [mod1 - Function Block Diagram]
File Edit View Algorithm FB Debug SFC Debug Tools Windows Help
All FB Items
PID
Ramp
Running Average
Rate Limit
Scaler
Signal Generator
Splitter
Subtract
Traveler
Period
Ratchet
Timer
RS
Ratio
SSCR
Signal Se
SR
Timer Pul
ExclusiveO
For Help, press F1
Assigned To: Controller1
ADD
ADD1
IN1
IN2
OUT
ATTR1
XFR
XFR1
SELECTION
BLOCK1
ATTR2
ATTR3
TP
TP1
IN_D
OUT_D
60
62
64
66
68
68
70

FIG. 2

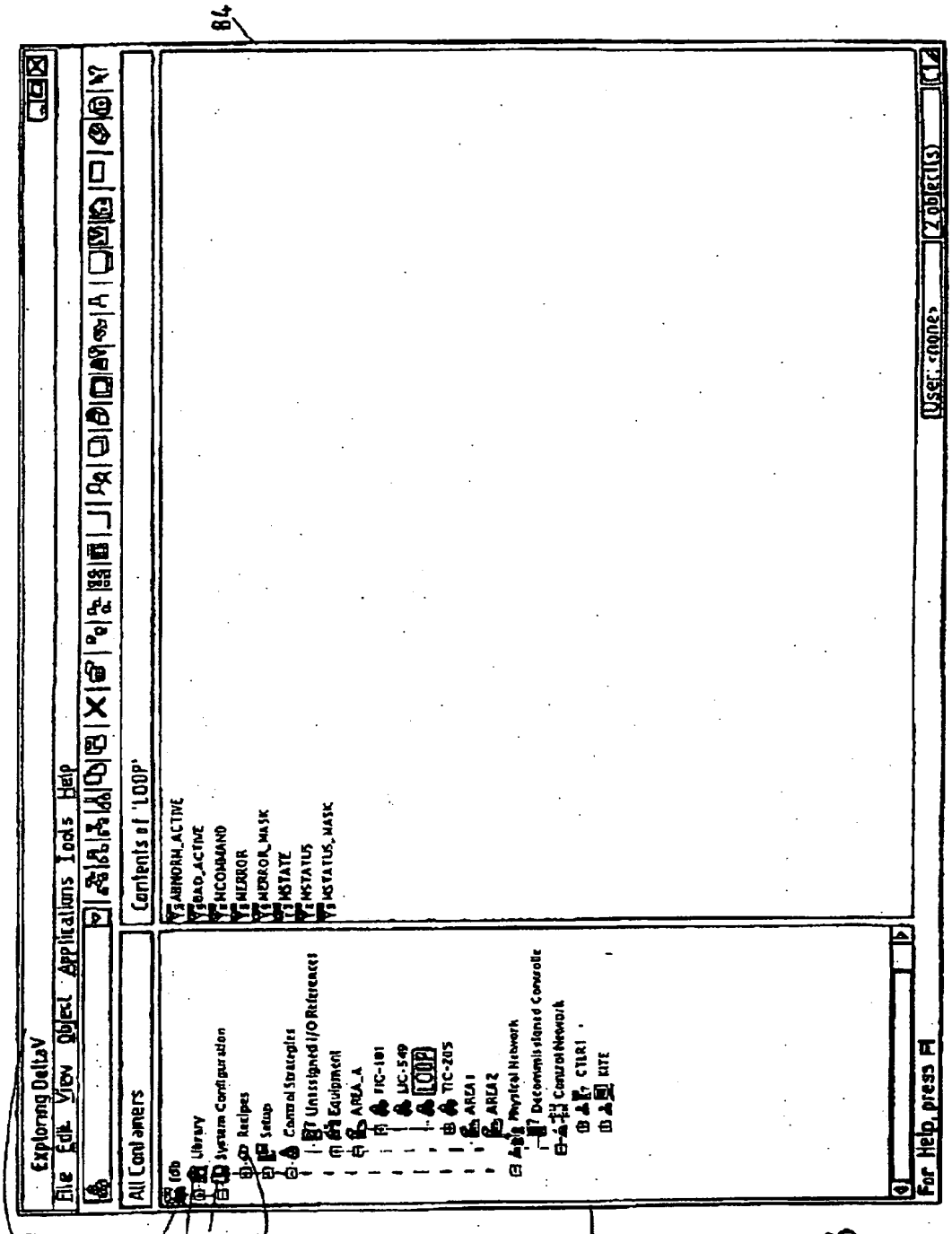

Exploring DeltaV
File Edit View Object Applications Tools Help
All Containers
Contents of 'LOOP'
Library
System Configuration
Recipes
Setup
Control Strategies
Unassigned I/O References
Equipment
AREA_A
FIC-101
LIC-549
LOOP
TIC-205
AREA 1
AREA 2
Physical Network
Decommissioned Controllc
ControlNetwork
CTLR1
RTE
ABNORM_ACTIVE
BAD_ACTIVE
HCOMMAND
MERROR
MERROR_MASK
MSTATE
MSTATUS
MSTATUS_MASK
For Help, press F1
User: <none>
2 object(s)
80
82
84
86
88
90
92

FIG. 3

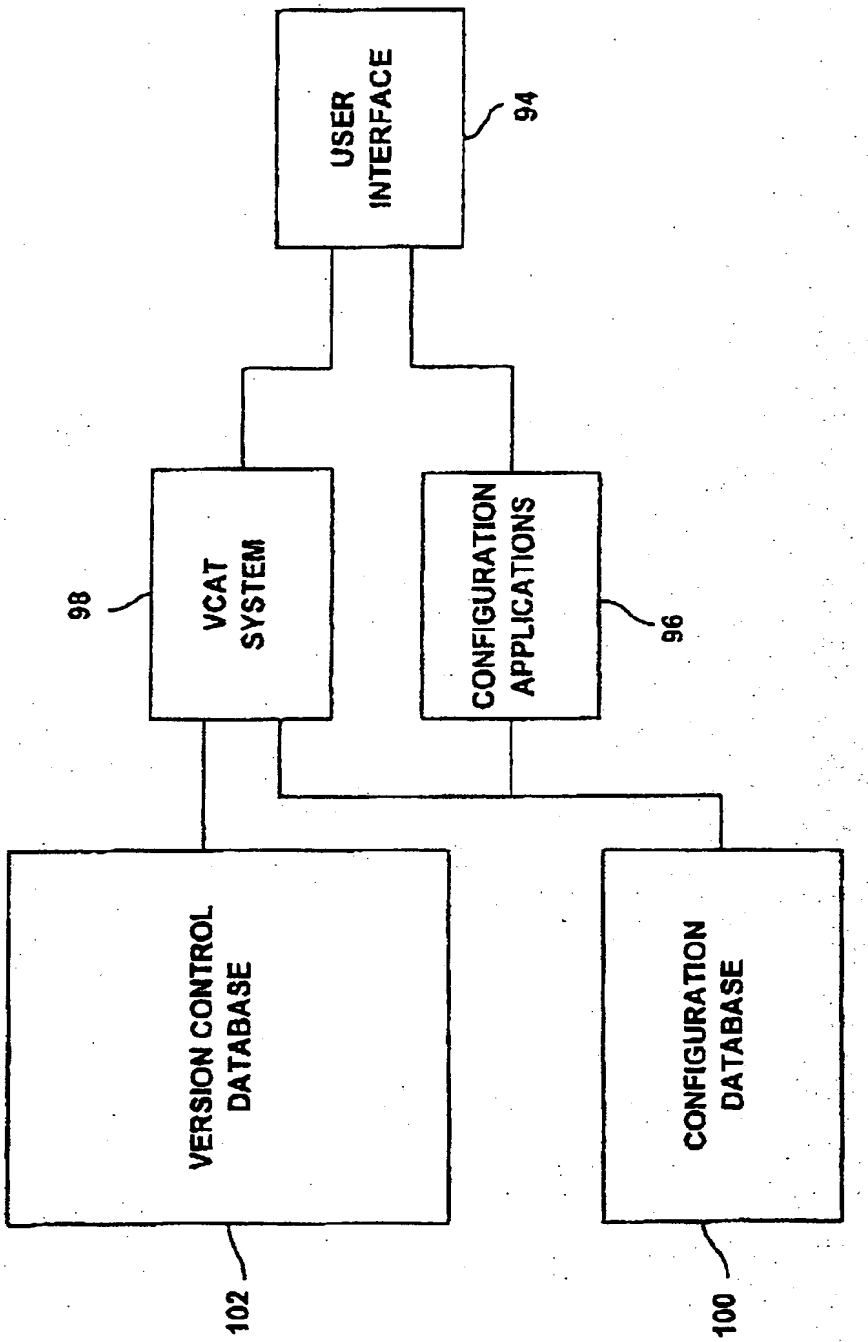

USER INTERFACE
94
VCAT SYSTEM
98
CONFIGURATION APPLICATIONS
96
VERSION CONTROL DATABASE
102
CONFIGURATION DATABASE
100

FIG. 4

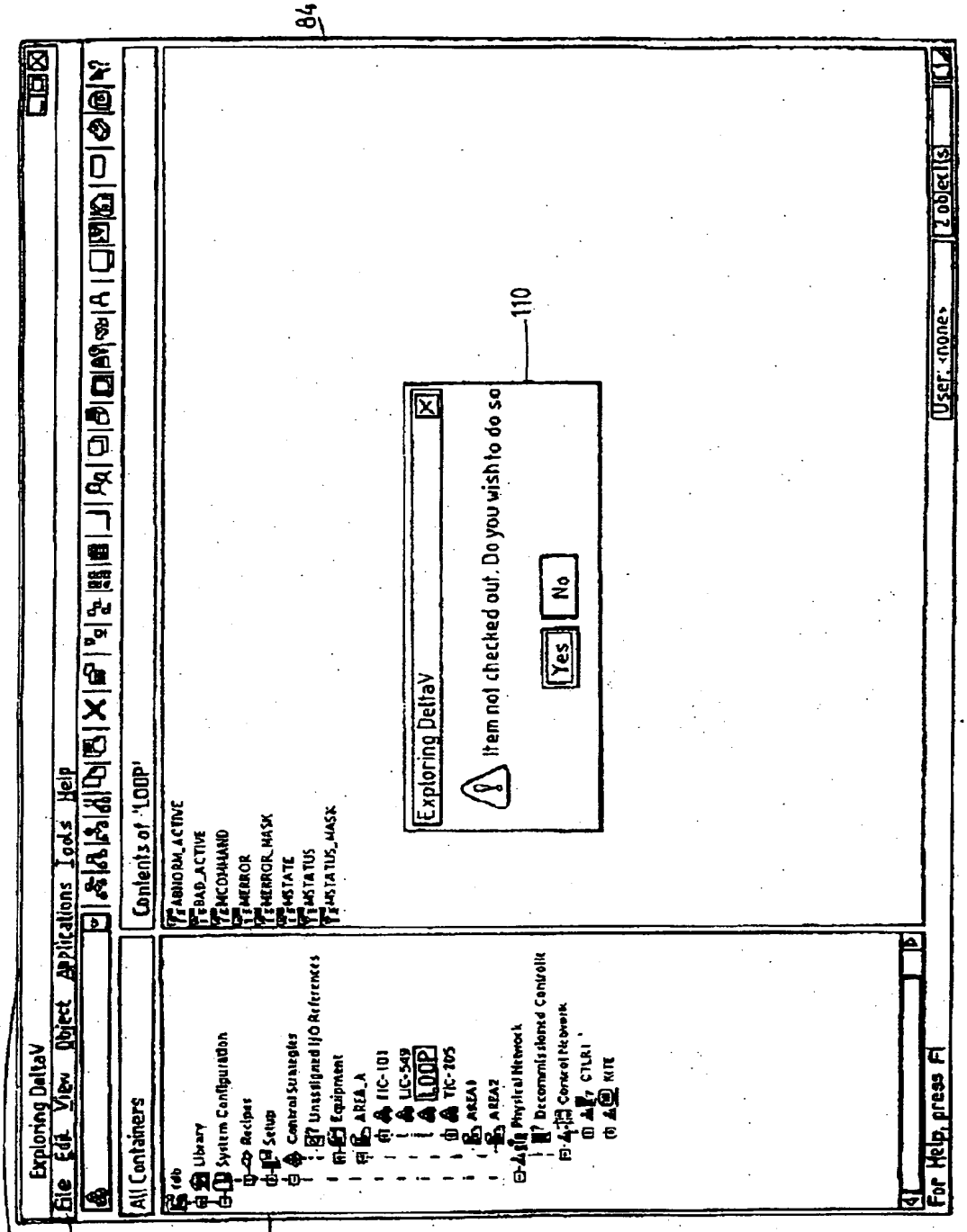

80
112
82
84
110
Exploring DeltaV
File Edit View Object Applications Tools Help
All Containers
Contents of 'LOOP'
Exploring DeltaV
Item not checked out. Do you wish to do so
Yes
No
For Help, press F1
User: <none>
2 object(s)

FIG. 5

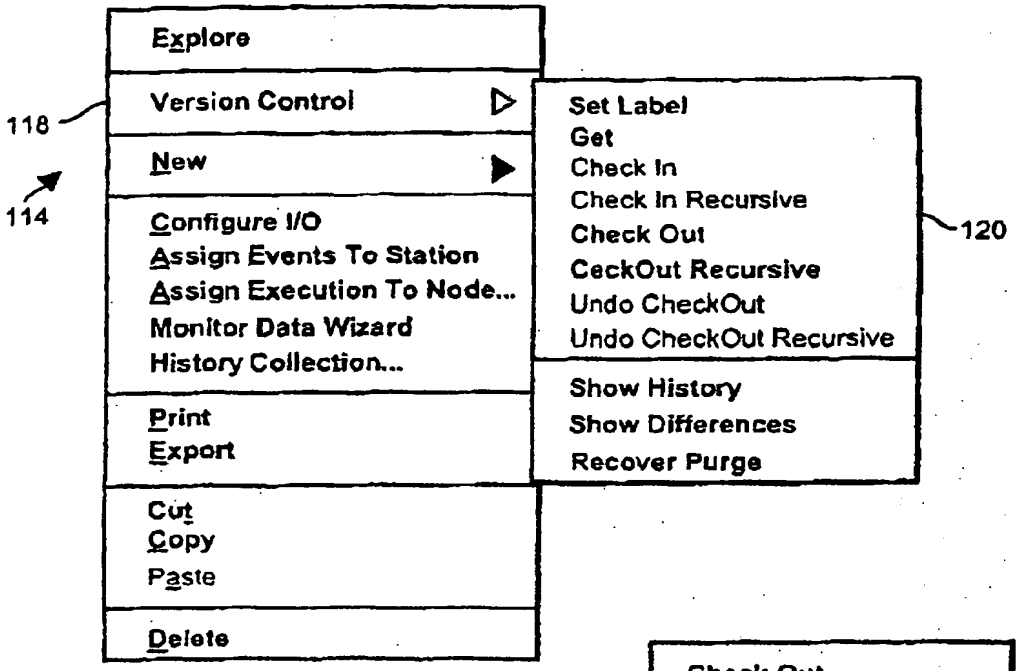


FIG. 6

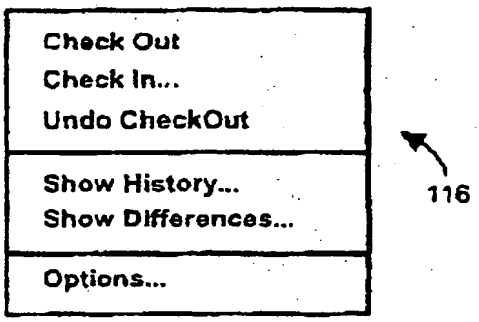


FIG. 7

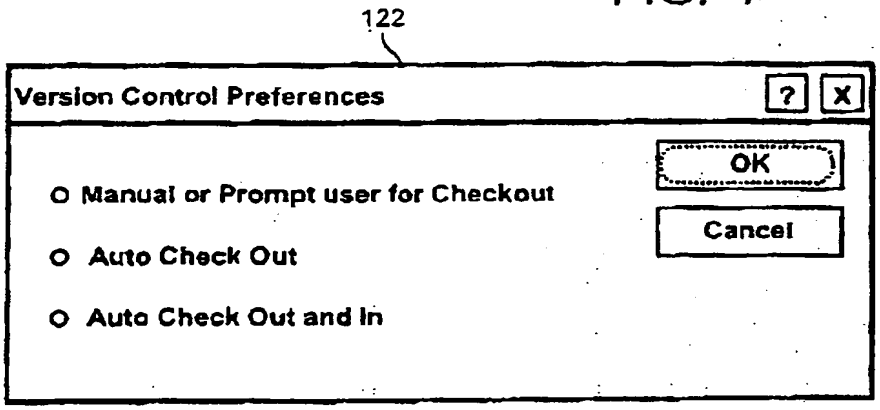


FIG. 8

FIG. 9

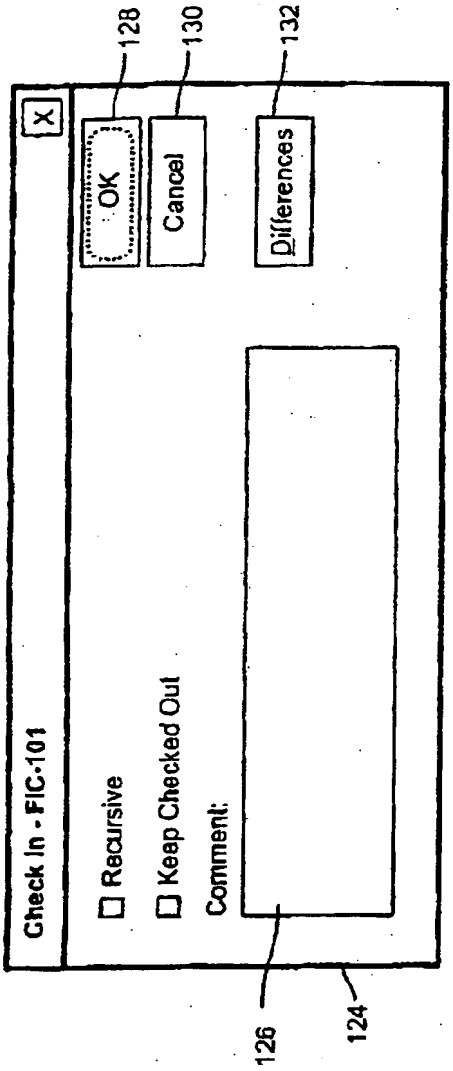
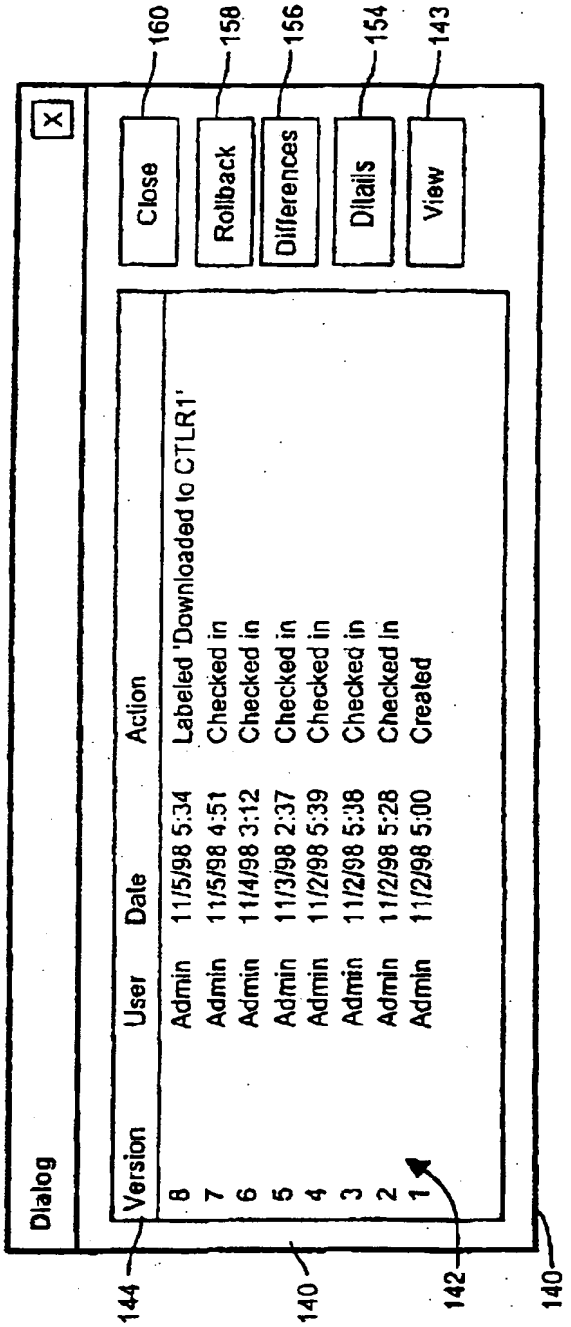

Check In - FIC-101
X
Recursive
Keep Checked Out
Comment:
OK
Cancel
Differences
128
130
132
126
124

FIG. 10


Dialog
X
Close
Rollback
Differences
Dilails
View
160
158
156
154
143
Version
User
Date
Action
8 Admin 11/5/98 5:34 Labeled 'Downloaded to CTLR1'
7 Admin 11/5/98 4:51 Checked in
6 Admin 11/4/98 3:12 Checked in
5 Admin 11/3/98 2:37 Checked in
4 Admin 11/2/98 5:39 Checked in
3 Admin 11/2/98 5:38 Checked in
2 Admin 11/2/98 5:28 Checked in
1 Admin 11/2/98 5:00 Created
144
140
142
140

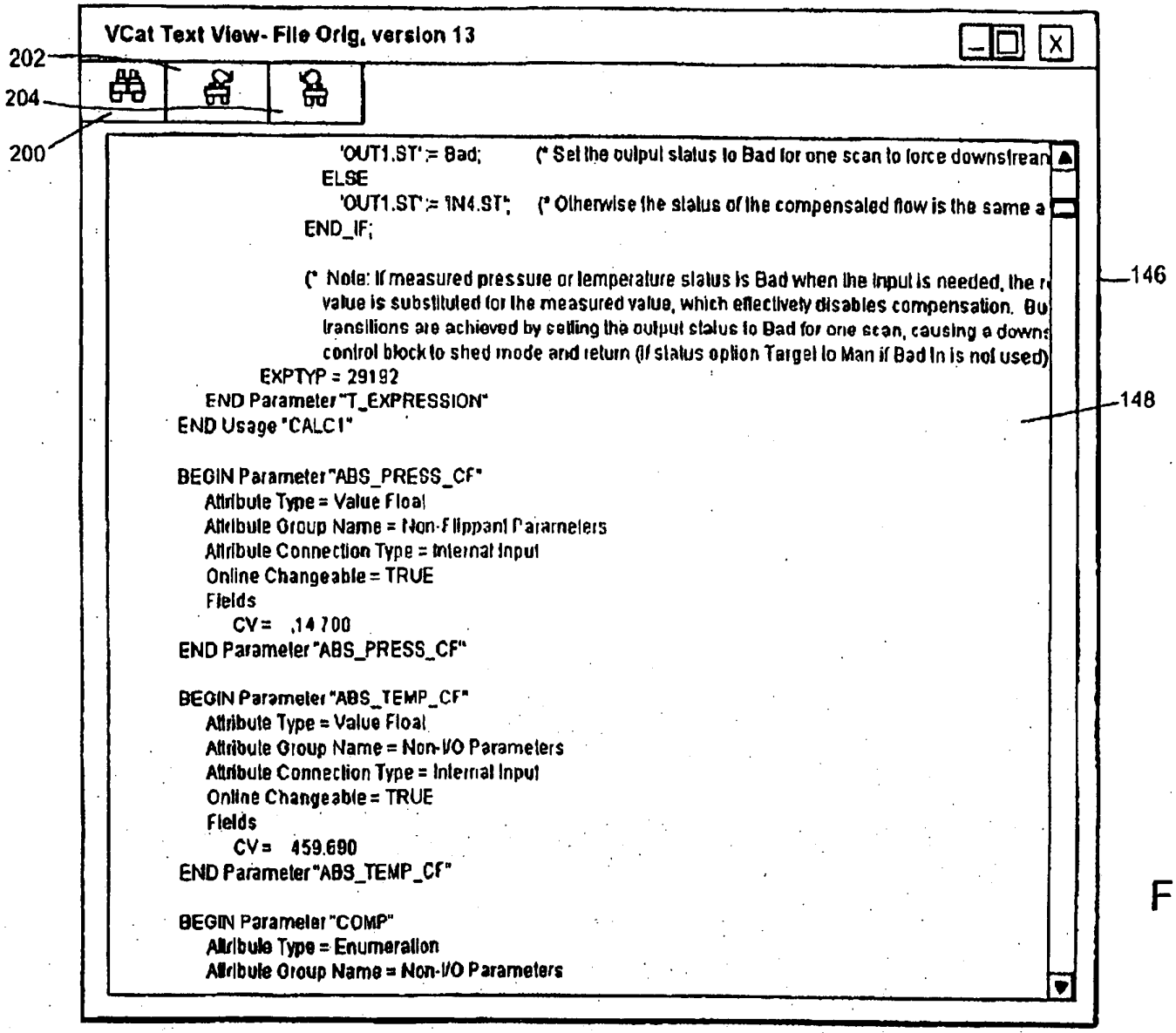
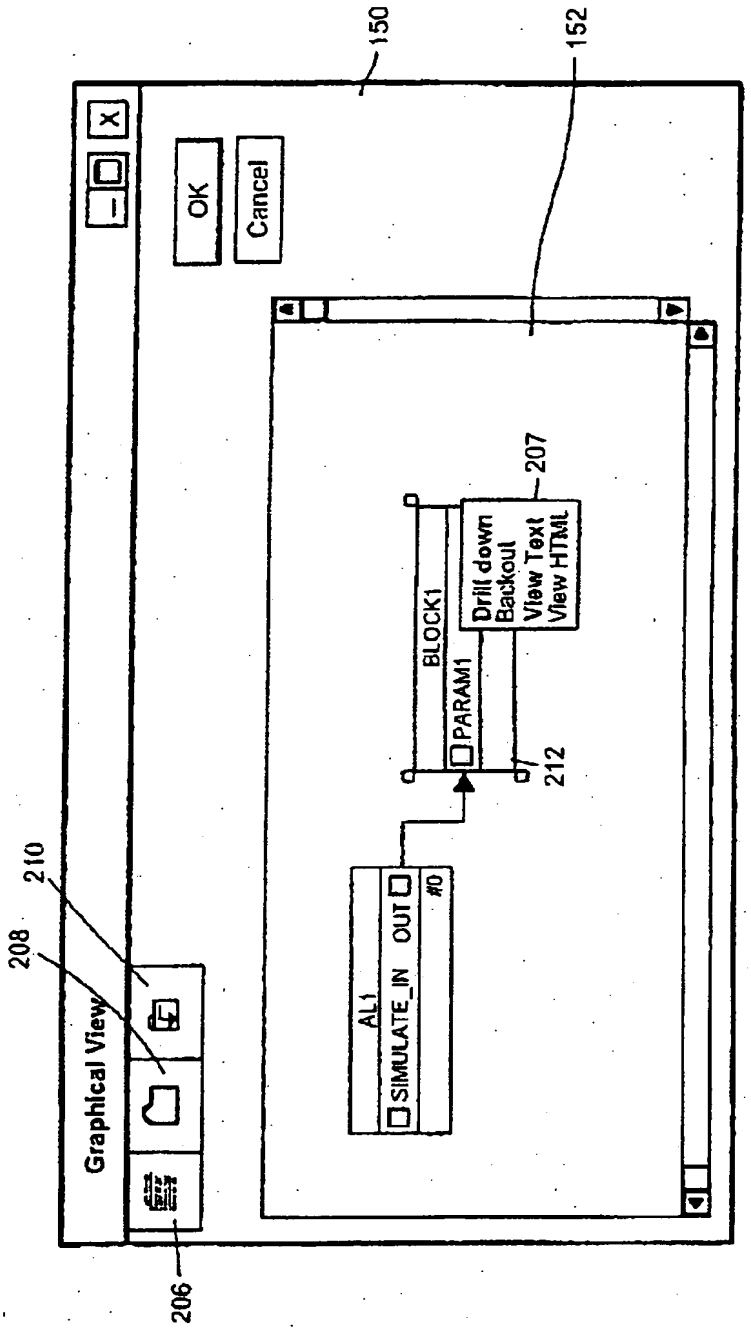

VCat Text View- File Orig, version 13
202
204
200
'OUT1.ST' := Bad; (* Set the output status to Bad for one scan to force downstrean
ELSE
'OUT1.ST' := IN4.ST; (* Otherwise the status of the compensated flow is the same a
END_IF;
(* Note: If measured pressure or temperature status is Bad when the input is needed, the r
value is substituted for the measured value, which effectively disables compensation. Bu
transitions are achieved by setting the output status to Bad for one scan, causing a downs
control block to shed mode and return (if status option Target to Man if Bad In is not used)
EXPTYP = 29192
END Parameter "T_EXPRESSION"
END Usage "CALC1"
146
148
BEGIN Parameter "ABS_PRESS_CF"
Attribute Type = Value Float
Attribute Group Name = Non-Flippant Parameters
Attribute Connection Type = Internal Input
Online Changeable = TRUE
Fields
CV = 14.700
END Parameter "ABS_PRESS_CF"
BEGIN Parameter "ABS_TEMP_CF"
Attribute Type = Value Float
Attribute Group Name = Non-I/O Parameters
Attribute Connection Type = Internal Input
Online Changeable = TRUE
Fields
CV = 459.690
END Parameter "ABS_TEMP_CF"
BEGIN Parameter "COMP"
Attribute Type = Enumeration
Attribute Group Name = Non-I/O Parameters

FIG. 11

FIG. 12


150
152
OK
Cancel
207
Drill down
Backout
View Text
View HTML
BLOCK1
PARAM1
212
AL1
SIMULATE_IN
OUT
#0
210
208
Graphical View
206

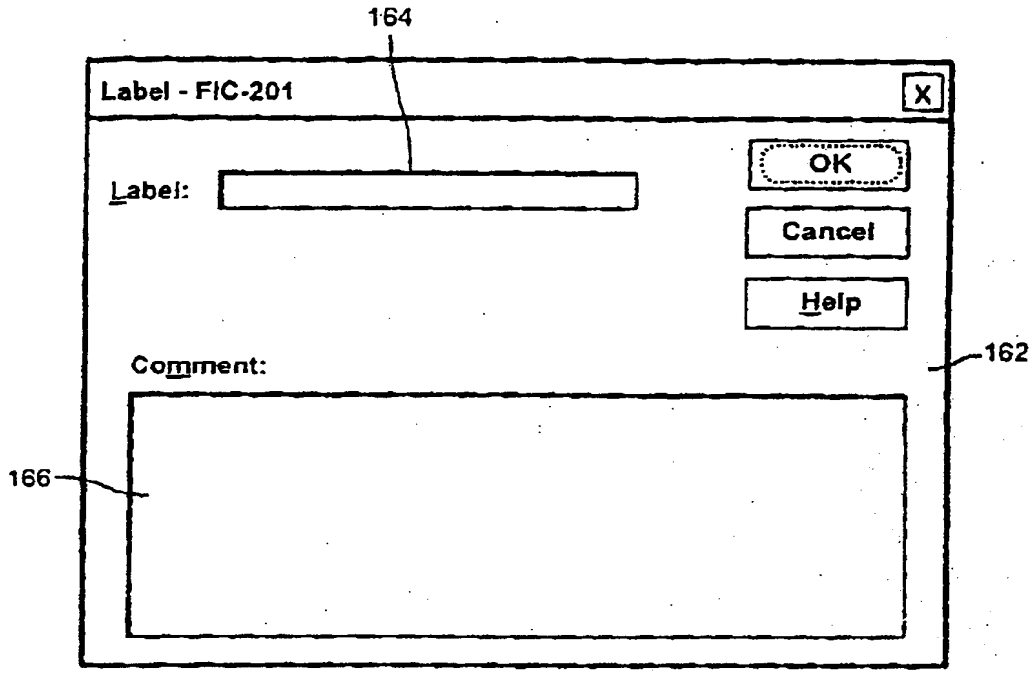


FIG. 13

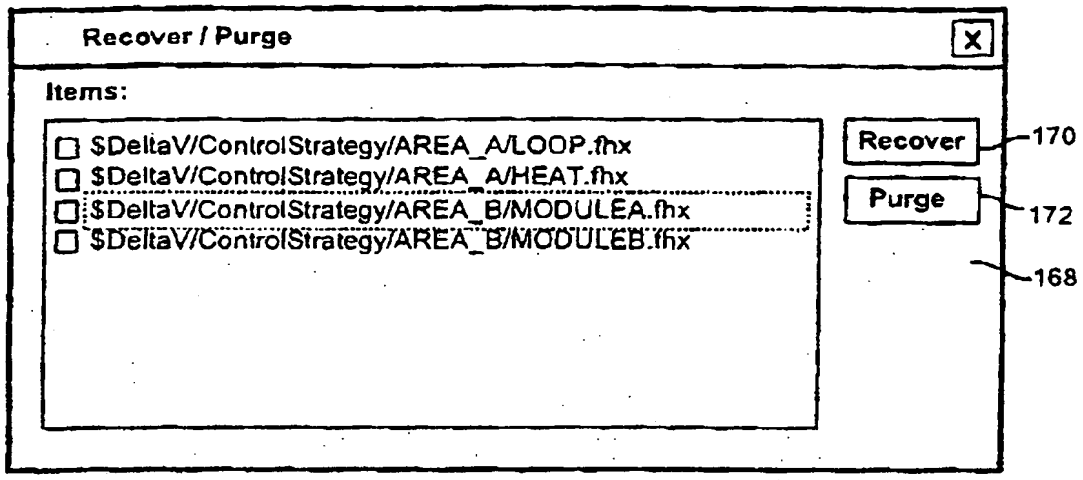

Recover / Purge
Items:
$DeltaV/ControlStrategy/AREA_A/LOOP.fhx
$DeltaV/ControlStrategy/AREA_A/HEAT.fhx
$DeltaV/ControlStrategy/AREA_B/MODULEA.fhx
$DeltaV/ControlStrategy/AREA_B/MODULEB.fhx
Recover
Purge
170
172
168

FIG. 14

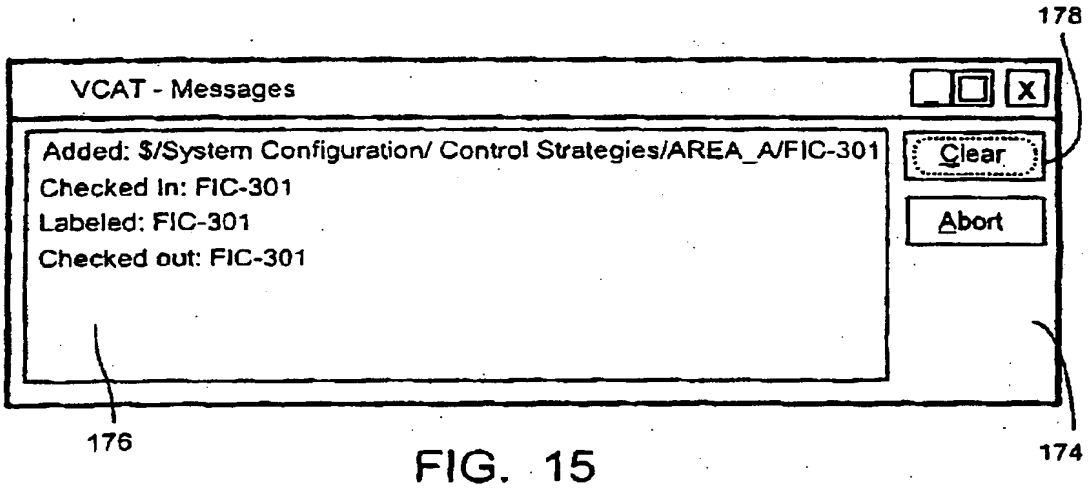

178
VCAT - Messages
Added: $/System Configuration/ Control Strategies/AREA_A/FIC-301
Checked In: FIC-301
Labeled: FIC-301
Checked out: FIC-301
Clear
Abort
176
174

FIG. 15

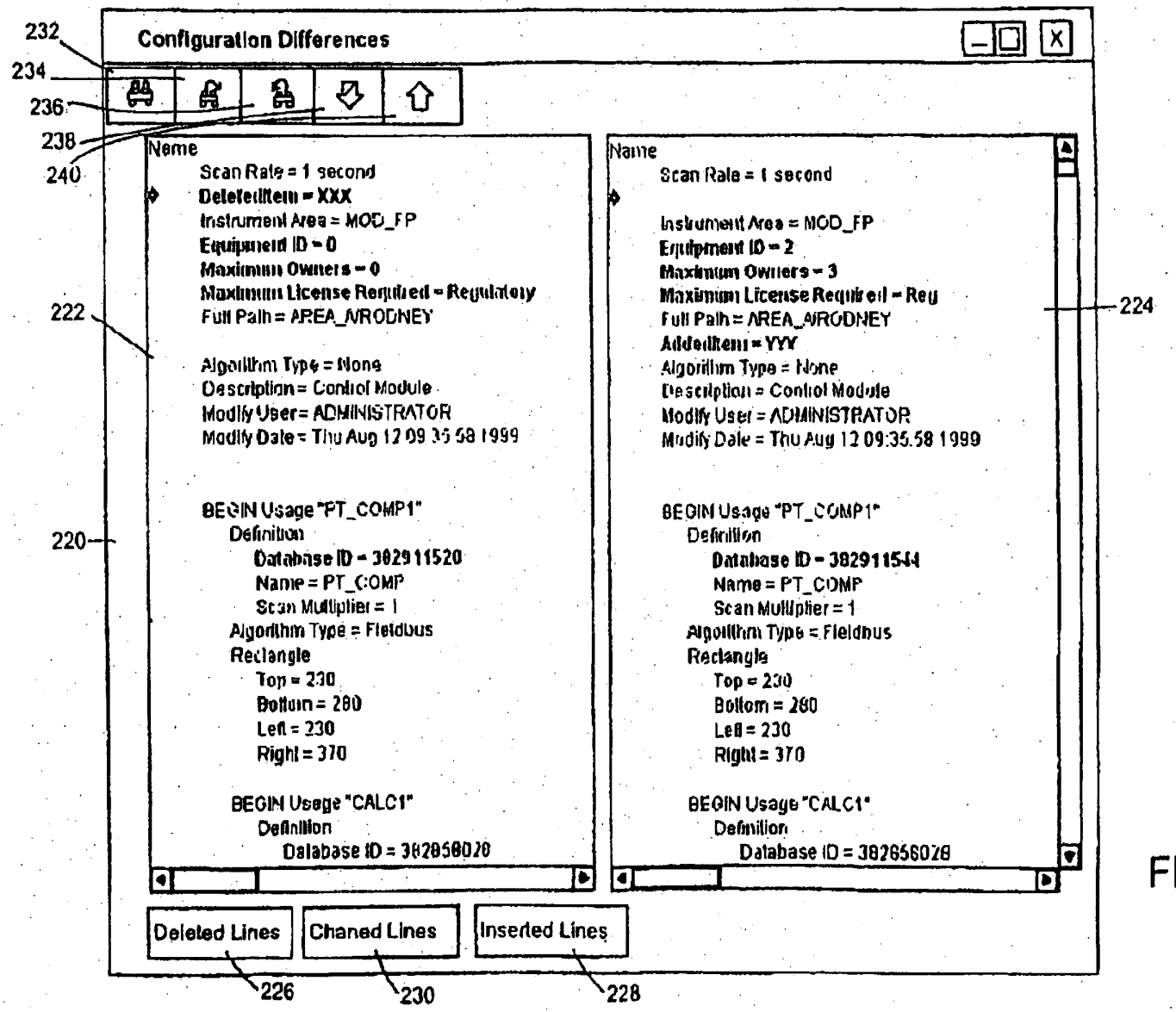
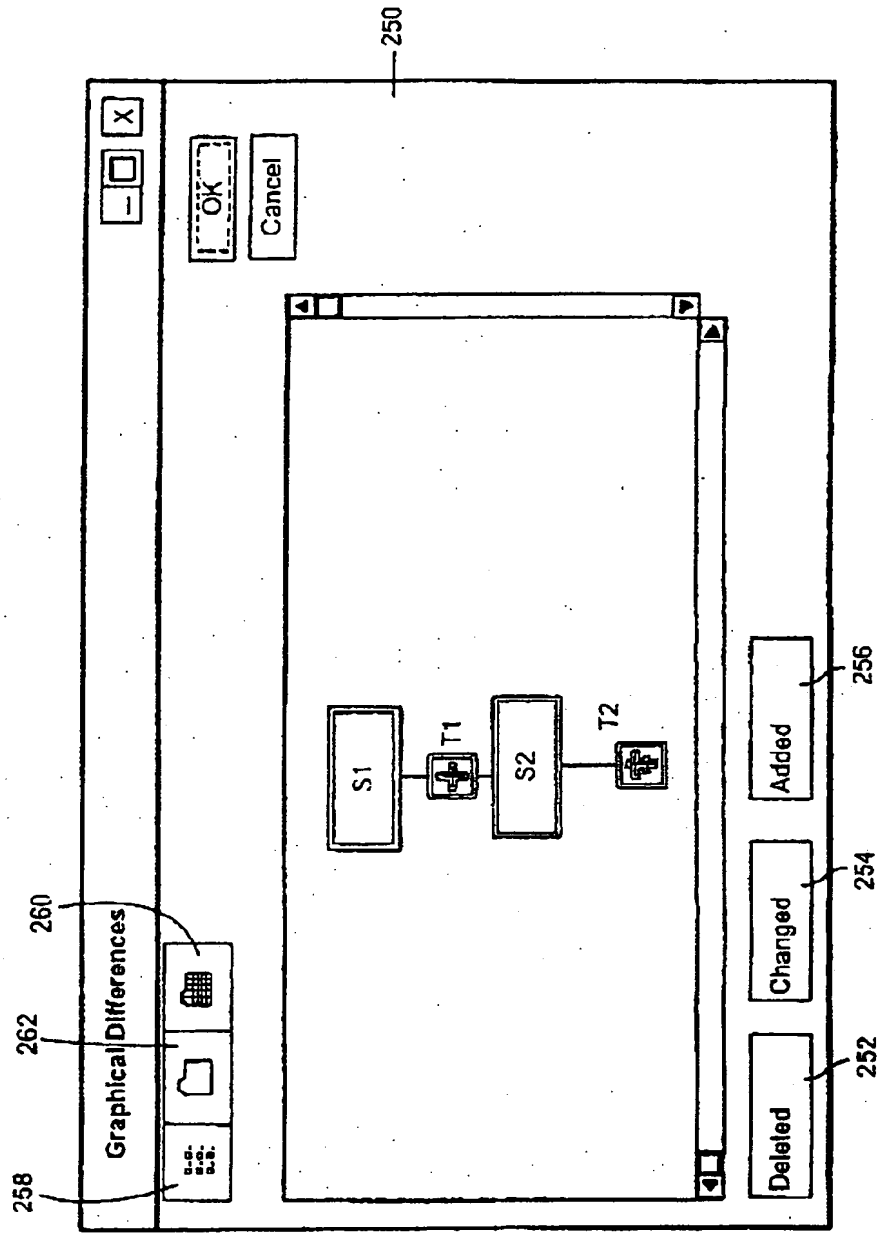

Configuration Differences
232
234
236
238
240
222
220
224
Name
Scan Rate = 1 second
DeletedItem = XXX
Instrument Area = MOD_FP
Equipment ID = 0
Maximum Owners = 0
Maximum License Required = Regulatory
Full Path = AREA_ARODNEY
Algorithm Type = None
Description = Control Module
Modify User = ADMINISTRATOR
Modify Date = Thu Aug 12 09:35:58 1999
BEGIN Usage "PT_COMP1"
Definition
Database ID = 382911520
Name = PT_COMP
Scan Multiplier = 1
Algorithm Type = Fieldbus
Rectangle
Top = 230
Bottom = 280
Left = 230
Right = 370
BEGIN Usage "CALC1"
Definition
Database ID = 382656028
Name
Scan Rate = 1 second
Instrument Area = MOD_FP
Equipment ID = 2
Maximum Owners = 3
Maximum License Required = Reg
Full Path = AREA_ARODNEY
AddedItem = YYY
Algorithm Type = None
Description = Control Module
Modify User = ADMINISTRATOR
Modify Date = Thu Aug 12 09:35:58 1999
BEGIN Usage "PT_COMP1"
Definition
Database ID = 382911544
Name = PT_COMP
Scan Multiplier = 1
Algorithm Type = Fieldbus
Rectangle
Top = 230
Bottom = 280
Left = 230
Right = 370
BEGIN Usage "CALC1"
Definition
Database ID = 382656028
Deleted Lines
Changed Lines
Inserted Lines
226
230
228

FIG. 16

FIG. 17


250
OK
Cancel
S1
T1
S2
T2
Added
256
Changed
254
Deleted
252
Graphical Differences
260
262
258

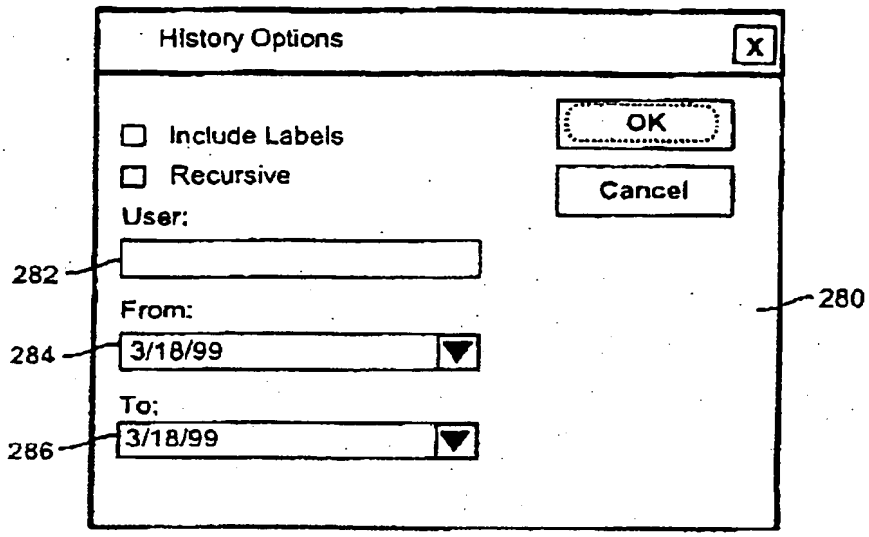

History Options
X
Include Labels
Recursive
User:
OK
Cancel
282
280
From:
284
3/18/99
To:
286
3/18/99

FIG. 18

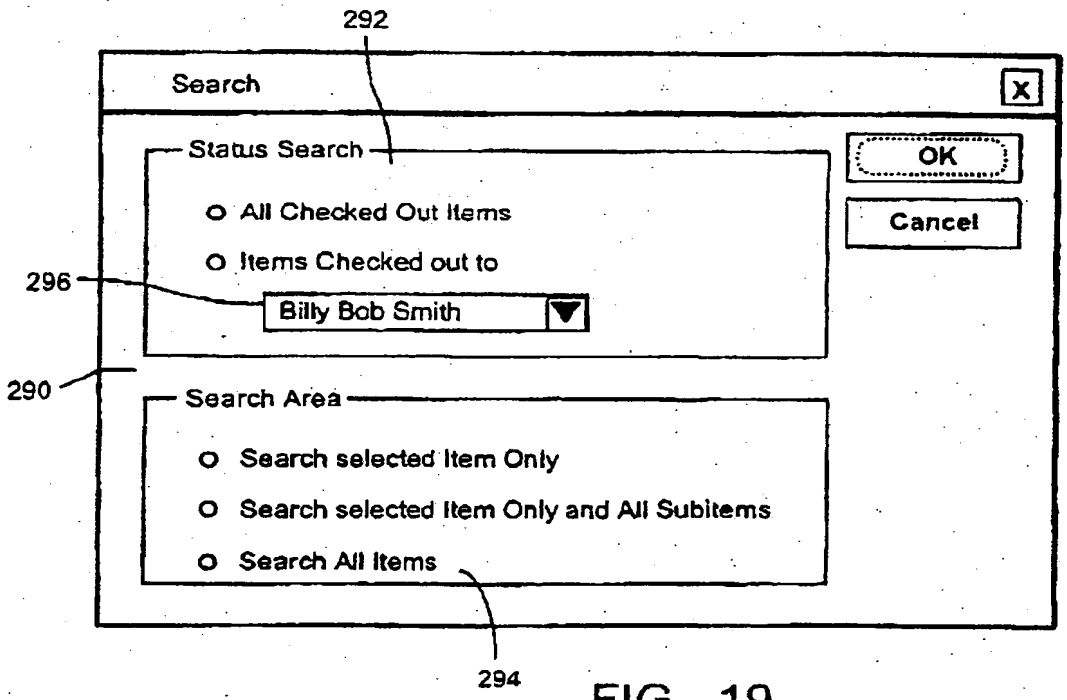

292
Search
X
Status Search
OK
All Checked Out Items
Items Checked out to
Cancel
296
Billy Bob Smith
290
Search Area
Search selected Item Only
Search selected Item Only and All Subitems
Search All Items
294

FIG. 19

1 Abstract

[Means of Resolving]

A system useful for controlling a process includes a computer-readable medium and a processor in communication with the computer-readable medium. The system further includes a first database and a second database. The first database stores first data representative of a first configuration of the process, while the second database stores second data representative of a second configuration of the process. A configuration routine of the system is stored in the computer-readable medium and configured to be executed by the processor to facilitate a modification of the first configuration of the process. A version control routine of the system is stored in the computer-readable medium and configured to be executed by the processor to store in the second database third data indicative of the modification of the first configuration of the process.

2 Representative Drawings

Fig. 4